GEOMETRIC, FEATURE-BASED AND GRAPH-BASED APPROACHES FOR THE STRUCTURAL ANALYSIS OF PROTEIN BINDING SITES:

NOVEL METHODS AND COMPUTATIONAL ANALYSIS

Dissertation

zur Erlangung des Doktorgrades
der Naturwissenschaften
(Dr. rer. nat.)

dem Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg
vorgelegt

von

Thomas Fober

Marburg, 2013

Vom Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg als Dissertation
angenommen am: 03.04.2013

Erstgutachter: Prof. Dr. Eyke Hüllermeier
Zweitgutachter: Prof. Dr. Gerhard Klebe

weitere Mitglieder der Prüfungskommission:

Prof. Dr. Bernd Freisleben
Prof. Dr. Alfred Ultsch

Tag der mündlichen Prüfung: 08.04.2013

Meinen Eltern

Contents

List of Figures

List of Tables

Acknowledgment

After starting as entrant at Philipps-Universität Marburg, I had to learn several things, among others, how to tackle research problems, how to write scientific papers, how to prepare a talk and how to hold it. In all these areas, Prof. Hüllermeier was and still is an exceptionally gifted mentor and I thank him for his enormous guidance which helped me not only during my time in Marburg, but which — as I am convinced — will also proofs beneficial for my future. I appreciate the pleasant working atmosphere and his continuous support, moreover, that he always found time to listen to any kind of problems. Without doubts, his invaluable assistance led finally to the success of this thesis and the immense progress we made in the last 5 years in the structural bioinformatics domain. Last but not least, I thank him for giving me the opportunity to work in his group and for providing very interesting and diversified research topics. Thanks a lot, Eyke!

Prof. Klebe I thank for reviewing this thesis from the pharmaceutical focus, and for the helpful comments on the datasets we used and the results we obtained. Moreover I thank him a lot for enabling my visit to *The Cambridge Crystallographic Data Centre* in January 2012.

I thank all my colleagues in the mathematics and computer science department and the department of pharmaceutical chemistry, in particular Serghei Glinca, Timo Krotzky, Thomas Rickmeyer, Yu Yi, Florian Meyer and Marc Strickert. Serghei Glinca, Timo Krotzky and Thomas Rickmeyer for the discussions related to CavBase, proteins and protein binding sites; Yu Yi for sharing his fantastic programming skills with me; Florian Meyer for the helpful discussions on metrics and the programming language R; finally Marc Strickert, who joint our group last year, for the gainful and pleasant cooperation in analyzing large datasets. Among my colleagues, the largest gratitude goes to my office mate Marco Mernberger for creating an enjoyable work environment and for the excellent cooperation we had.

I also thank the *The Cambridge Crystallographic Data Centre* staff for the enjoyable time I had during my research stay, for sharing all information — even the CavBase source code — with me and for the very helpful discussions. In

this regards, I thank especially Simon Cottrell who always found some time to answer my question regarding the implementation of CavBase.

Summary

With the introduction of databases in biology, computer science became part of biology and the new research field called bioinformatics emerged. Bioinformatics led without doubts to a break-through in biology and was so far dominated by methods on sequences, e.g. sequences of amino acids. A high similarity on sequence level is usually related with a high similarity on the structural level, however, the other direction does not hold. Hence, similar structures may not have similar sequences. The prediction of protein function on the sequence level is therefore not optimal since function is more related to structure than to sequence. Due to the continually increasing number of entries in structural protein databases it becomes, however, possible to perform calculations directly on the structural level which usually leads to much more reliable results.

In the past years a lot of work was presented for function prediction on structural level, often making use of additional information, as the fold of a protein. This additional information allows to process on more compact models of the protein based, e.g., on C_α atoms leading to an acceptable runtime. CavBase realizes another approach: Motivated by the fact that the active site of a protein is mainly responsible for its function, in CavBase only protein binding sites are stored and processed. By using further abstraction a compact set of 3-dimensional descriptors is generated which allow for more efficient calculations. In CavBase these calculations are based on subgraph isomorphism which requires to transform the data into graphs.

Subgraph isomorphism is a very intuitive concept calculating a partial alignment which can be used to easily derive a similarity measure. Unfortunately, this concepts leads also to some problems: Finding the maximum common subgraph of two graphs is an NP-complete problem, usually solved by searching for the maximum clique in the product graph. Moreover, only a very small degree of error-tolerance is obtained, which poses a problem if this measure is applied on real-world applications where data is subject to noise, structural deformations and mutations. Another problem is caused by the required transformation of the data into graphs, a procedure which is usually leading to a loss of information making a post-processing necessary. Due to these reasons, a large-scale comparison of protein binding sites may become interminable, moreover, it is even not guaranteed that a single comparison can

be calculated since the required product graph can become too large.

Beside the calculation of similarity often (multiple) alignments are required. So far calculation of multiple alignments in CavBase is realized by a greedy extension of the partial alignment found by applying clique detection and the subsequent execution of the star-alignment technique. This approach combines both, the disadvantages of an exact algorithm and those of two greedy heuristics: On non matroids, greedy heuristics lead to suboptimal solutions, exact algorithms, on the other hand, lead to exponential runtime when NP-hard problems are considered. Hence, the resulting approach is neither efficient nor exact.

Despite all problems, one should not dismiss the analysis of proteins on the structural level since it could be already shown that not all similarities can be detected on the sequence level alone. In this thesis, new algorithms are developed with the goal to increase efficiency in terms of runtime as well as space. A first step to reach this goal is to replace the exact clique search by an heuristic. The resulting approach still calculates a partial alignment, however, in polynomial time. For the calculation of similarity it is obviously not compulsory to establish such an alignment which is usually much harder to calculate as raw similarity. So-called feature-based approaches bear on the idea, that objects can be represented by vectors which can be compared subsequently. Such approaches will be much more efficient compared to approaches based on subgraph isomorphism. Instead of using feature-based methods, one can apply the R-convolution framework which can be used to define graph kernels. If one wants to calculate an alignment in a more error-tolerant way, the maximum common subgraph can be replaced by an approximate common subgraph. Technically, the search for cliques in the product graph is then replaced by a search for quasi-cliques. The highest degree on error-tolerance, however, is offered by methods which are based on the graph edit distance. Since the graph edit distance can be formulated as a combinatorial optimization problem, an evolutionary algorithm is employed to solve it. This algorithm will not be the most efficient one, however, it is very likely that the quality of the calculated edit operations is much higher compared to other heuristics. Finally, to enable processing on the raw CavBase data another class of algorithms is developed which does not cause a loss of information. Methods belonging to this class process on labeled point clouds which lead to algorithmic problems which exhibit some nice properties, as continuity, allowing for the calculation

of similarity or alignments in an efficient way. The main results can be summarized as follows:

- The widely used maximum common subgraph measure leads to surprisingly good results and can be applied easily since it does not require parameterization. Moreover, this measure can be relaxed in different ways, e.g., by using an appropriate normalization or the approximative maximum common subgraph. An very efficient approximation of the maximum common subgraph can be calculated as well, leading to a very efficient though not optimal approach.

- Even though approaches based on the graph edit distance are easy to understand, they are hard to parameterize. Moreover, the search space is exorbitant large making the search for the optimal solution difficult and very inefficient. However, for the calculation of multiple graph alignments this measure is needed. In this thesis it turned out that especially the evolutionary algorithm lead to much better results than its counterpart based on a greedy heuristic.

- Graph kernels are similarity measures on graphs which can be computed efficiently. Unfortunately, it turned out that they are inappropriate for the comparison of protein structures, in particular because more efficient feature-based approaches perform much better in all executed experiments.

- CavBase data can be processed directly, without transforming it into graphs. For this purpose labeled point clouds are introduced in this thesis which can be used to calculated similarity and multiple alignments as well. The thus constructed alignments exhibit a high quality, the similarity measure used for classification and retrieval led to high accuracies and good rankings. A main advantage of the developed approaches is their low space complexity and their excellent scaling.

- Is the construction of alignments not required, feature-based approach allow for a very efficient calculation of similarity. Unfortunately, this class of approaches leads to the highest degree of loss of information since a vector representation of a protein binding site does not allow to reconstruct the binding site. Accordingly, such approaches perform well especially on smaller datasets. In the case of larger datasets the probability for

mapping dissimilar protein binding sites onto similar vectors increases, hence, the performance of these measures drops strongly. Anyhow, since the converse argument does not hold, i.e., protein binding sites which are mapped onto dissimilar vectors are dissimilar, feature-based approaches can be used to identify dissimilar binding sites which can be removed afterwards. This would lead to a speed-up especially during execution of large scale studies.

Zusammenfassung

Mit der Einführung von Datenbanken in die Biologie, ist die Informatik auch in das Fachgebiet der Biologie geschritten und eine neue Fachrichtung, die Bioinformatik, wurde begründet. In den letzten Jahrzehnten war die Bioinformatik vor allem durch das Sequenzalignment und weiteren Methoden auf Sequenzen geprägt, welche zweifelslos zu einem Durchbruch in der Biologie geführt haben. Eine hohe Ähnlichkeit auf Sequenzebene ist zwar in der Regel mit einer hohen Ähnlichkeit auf Strukturebene verbunden, jedoch müssen im Umkehrschluss ähnliche Strukturen nicht notwendigerweise mit ähnlichen Sequenzen einhergehen. Dies führt dazu, dass Verfahren basierend auf Sequenzen nicht die beste Wahl für die Funktionsvorhersage von Proteinen sind, da die Funktion eines Proteins wesentlich durch seine Struktur bestimmt wird. Für die Funktionsvorhersage sind daher Methoden die direkt auf der Struktur eines Proteins ansetzen sicherlich die bessere Wahl. Mit der stetig steigenden Zahl von Datenbankeinträgen über Struktur von Proteinen drängt es sich daher geradezu auf, direkt auf diesen Daten zu arbeiten und hierfür entsprechende Algorithmen zu entwickeln.

In den letzten Jahren sind auf dem Gebiet der Strukturanalyse bereits eindrucksvolle Arbeiten geleistet worden, insbesondere in Fällen, in denen auf weitere Informationen zurückgegriffen werden konnte, wie beispielsweise auf die Faltung eines Proteins. Verfahren für die reine Strukturanalyse, also insbesondere unabhängig von Sequenz und Faltung existieren zwar ebenfalls, sind allerdings geprägt von hohen Laufzeiten und einem hohen Speicherbedarf, der zu einem Scheitern der Verfahren auf vielen Eingaben führt. Damit sind solche Verfahren weit davon entfernt, einen Vergleich oder gar die Analyse einer ganzen Datenbank zu ermöglichen. In der Bioinformatik wird daher oft eine kompaktere Repräsentation gewählt, in der z.B. nur C_α Atome betrachtet werden. Ein anderer Weg wurde in der Datenbank CavBase gewählt: Motiviert durch die Tatsache, dass die Funktion eines Proteins nicht durch seine gesamte Struktur bestimmt wird, sondern vielmehr durch die Geometrie und die physikochemischen Eigenschaften des aktiven Zentrums, werden in CavBase anstelle gesamter Proteine nur Proteinbindetaschen betrachtet. Durch weitere Abstraktionsschritte wird zudem eine eher kleine Menge an 3-dimensionalen Deskriptoren erzeugt, die dann eine effizientere Berechnung zulassen. Berechnungen in CavBase basieren dabei immer auf dem Konzept des Subgraph Isomorphis-

mus, das offensichtlich die Transformation der Ausgangsdaten in Graphen erforderlich macht.

Subgraph Isomorphismus ist ein sehr intuitives Konzept und erlaubt es sowohl ein Ähnlichkeitsmaß abzuleiten, als auch partielle Alignments zu berechnen. Leider führt dieses Konzept aber auch zu zahlreichen Problemen: Zum einen ist das Problem der Berechnung des maximalen Subgraphen zweier Graphen NP-vollständig und wird zumeist durch das Suchen einer Clique im Produktgraphen gelöst. Zum anderen erlaubt es nur einen sehr kleinen Grad an Fehlertoleranz, obwohl bei der Betrachtung biologischer Daten Messfehler, strukturelle Verformungen und Mutationen eintreten können, die die Geometrie oder die physikochemischen Eigenschaften zu einem gewissen Grad verändern. Des Weiteren müssen die Daten, die ursprünglich nicht in Form von Graphen gegeben sind, zunächst transformiert werden. Diese Transformation ist im allgemeinen Fall leider mit einem Informationsverlust behaftet und macht somit unter Umständen ein weiteres Post-Processing der Resultate erforderlich. Aufgrund dieser Probleme scheitert in aller Regel der Vergleich großer Mengen von Daten, zudem ist auf Eingaben bestimmter Größe eine Berechnung erst gar nicht möglich, da der Produktgraph eine enorme Größe annimmt und sich somit nicht ohne weiteres speichern lässt.

Neben paarweisen Vergleichen werden oft auch Methoden für die Berechnung (multipler) Alignments benötigt. Bisher ist dies in CavBase dadurch realisiert, dass die partiellen Alignments mittels einer greedy Heuristik zu vollständigen Alignments erweitert werden. Diese vollständigen paarweisen Alignments werden dann durch Anwendung der Stern-Alignment Methode zu einem multiplen Alignment verknüpft. Dieser Ansatz kombiniert offensichtlich die Nachteile eines exakten Verfahrens mit denen zweier greedy Heuristiken: Die Verwendung von greedy Heuristiken auf nicht-Matroiden führt zu suboptimalen Ergebnissen, wohingegen die Anwendung eines exakten Algorithmus auf dem NP-harten Problem *maximum clique* mit exponentieller Laufzeit verbunden ist. Somit ist ein solches Verfahren weder effizient noch exakt.

Trotz aller Schwierigkeiten sollte dennoch auf die Strukturanalyse nicht verzichtet werden, da bereits gezeigt werden konnte, dass Sequenzverfahren nicht alle Ähnlichkeiten detektieren können. Aus diesem Grund werden hier neue Verfahren vorgestellt, die den Vergleich und die Analyse von Proteinbindetaschen in einer effizienteren und fehlertoleranteren Weise erlauben. Diese Arbeit verfolgt daher mehrere Ziele: Es sollen Verfahren entwickelt werden,

die wesentlich weniger Ressourcen benötigen, im besten Fall sowohl effizienter sind als auch weniger Speicher konsumieren. Ein erster und einfacher Ansatz zur Effizienzsteigerung ist der Austausch der exakten Clique Suche durch eine Heuristik. So wird weiterhin ein partielles Alignment berechnet, dass allerdings in polynomieller Zeit konstruiert werden kann. Allein für die Berechnung der Ähnlichkeit ist es aber offensichtlich nicht zwingend notwendig auf ein partielles Alignment zurückzugreifen, dessen Berechnung NP-hart ist. Stattdessen werden hier Verfahren eingeführt, die ohne eine solche eins-zu-eins Zuordnung von Elementen auskommen. So genannte merkmalsbasierte Verfahren beruhen auf der Idee, ein Objekt auf einen Vektor abzubilden und nachfolgend Maße auf Vektoren zu verwenden. Solch ein Verfahren wird in der Regel zu einer weitaus effizienteren Methode führen, als ein Verfahren basierend auf Subgraph Isomorphismus. Neben merkmalsbasierten Verfahren können aber auch Methoden aus dem maschinellen Lernen Anwendung finden. Hier sind vor allem Graph Kerne hervorzuheben, die sich in vielen Bereichen als leistungsstarke und effiziente Ähnlichkeitsmaße auf Graphen erwiesen haben. Soll ein stärkerer Grad an Fehlertoleranz ermöglicht werden, so kann das Konzept der Quasi-Cliquen verwendet werden, um ein flexibleres Ähnlichkeitsmaß basierend auf approximativen maximalen Subgraphen abzuleiten. Den höchsten Grad an Flexibilität bieten allerdings Methoden die auf der Graph-edit Distanz basieren. Da die Graph-edit Distanz ein kombinatorisches Optimierungsproblem darstellt, wird hier von einem evolutionären Algorithmus Gebrauch gemacht, der dieses Problem vermutlich nicht effizient lösen kann. Im Vergleich zu dem ursprünglichen Verfahren ist allerdings davon auszugehen, dass eine wesentlich höhere Lösungsgüte erzeugt wird. Neben Algorithmen, die auf der merkmals- oder graph-basierten Darstellung arbeiten, wurde eine weitere Algorithmenklasse entwickelt, die CavBase Daten direkt verarbeiten kann. Hier werden so genannte gelabelte Punktwolken verwendet, die es nicht mehr erforderlich machen, die Daten in Graphen oder Vektoren zu transformieren; somit garantieren solche Verfahren eine verlustfreie Weiterverarbeitung der Daten. Zudem weisen die resultierenden algorithmischen Probleme oft interessante Eigenschaften – wie beispielsweise Stetigkeit – auf, die eine effiziente Berechnung von Ähnlichkeit oder Alignments ermöglichen. Die Hauptergebnisse dieser Arbeit können wie folgt zusammengefasst werden:

- Das weitverbreitete, auf dem maximalen Subgraph basierende Ähnlichkeitsmaß auf Graphen liefert überraschend gute Ergebnisse und lässt sich besonders einfach anwenden, da es parameterfrei ist. Relaxierungen dieses Maßes können auf verschiedene Weise realisiert werden, unter anderem durch Verwednung einer anderen Normalisierung bzw. der Verwendung des approximativen größten Subgraphen, letzterer, der vor allem bei flexiblen Proteinklassen Vorteile aufweist. Darüber hinaus ist es möglich den maximalen gemeinsamen Subgraph effizient anzunähern, in dem eine einfache Heuristik angewendet wird.

- Maße basierend auf der Graph-edit Distanz sind zwar sehr intuitiv, ein zentrales Problem ist jedoch deren Parametrisierung die nur sehr schwer bestimmbar ist. Darüber hinaus ist der zugrunde liegende Suchraum exorbitant groß, die Suche nach der optimalen Editierdistanz somit ineffizient. Dennoch ist dieses Maß vor allem dann notwendig, wenn multiple Graphalignments berechnet werden sollen. Hier hat sich ganz klar gezeigt, dass der in dieser Arbeit vorgestellte evolutionäre Algorithmus zu deutlich besseren Ergebnissen führt als sein Pendant basierend auf einer greedy Heuristik.

- So genannte Graph-kernels sind Ähnlichkeitsmaße auf Graphen, die sich besonders effizient und leicht berechnen lassen. Leider sind diese Maße aber für den Vergleich von Proteinbindetaschen ungeeignet, da sie auf allen in dieser Arbeit durchgeführten Experimenten zu schlechteren Ergebnissen führen, als effizientere merkmalsbasierte Verfahren.

- CavBase Daten können auch direkt verarbeitet werden, also ohne den Umweg einer Transformation hin zu Graphen. Zu diesem Zweck wurden Punktwolken eingeführt auf denen sowohl ein multiples Alignment berechnet, sowie eine Ähnlichkeitsfunktion definiert werden kann. Die konstruierten multiplen Alignments weisen dabei eine sehr hohe Güte auf; auf Klassifikations- oder Retrievalproblemstellungen angewendet liefert das Ähnlichkeitsmaß auf der anderen Seite sehr gute Ergebnisse. Besonders positiv an den so genannten geometrischen Ansätzen ist, dass sie sehr gut skalieren und eine sehr geringe Speicherkomplexität aufweisen.

- Ist die Berechnung eines Alignments nicht notwendig, so erlauben es

merkmalsbasierte Verfahren sehr effizeint Ähnlichkeit zwischen Proteinbindetaschen zu berechnen. Leider führt diese Klasse von Verfahren aber auch zu dem höhsten Informationsverlust, da ausgehend von der Vektorrepräsentation die Proteintasche nicht mehr rekonstruiert werden kann. Entsprechend arbeiten diese Verfahren auch insbesondere auf kleinen Datensätzen gut. Bei Verwendung größerer Datensätze, bei denen die Wahrscheinlichkeit steigt, dass unähnliche Proteine auf ähnliche Vektoren abgebildet werden, liefern merkmalsbasierte Verfahren eher schlechte Ergebnisse. Da der Umkehrschluss allerdings nicht gilt, können diese Verfahren benutzt werden um unähnliche Bindetaschen zu detektieren und somit zu Beschleunigung bei der Durchführung großer Studien führen.

Part I

Foundations

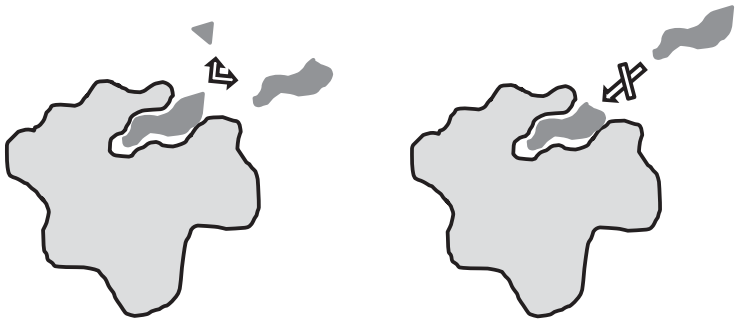
1

Introduction

Proteins are the elements of life and responsible for all functions in an organism, beginning with building of structure up to the regularization of chemical processes. Hence, for many scientific subjects proteins are very interesting objects. Here, proteins are considered from the chemogenomic (Bredel and Jacoby, 2004) point of view where the function of proteins is often of high interest (Pérot et al., 2010; Ekins, 2004; Vajda and Guarnieri, 2006; An et al., 2004; Weisel et al., 2009; Powers et al., 2006). Since the function of a protein is strongly related to its structure (Kinoshita et al., 2007; Watson et al., 2005; Kinoshita and Nakamura, 2003), and because high sequence similarity implies structural similarity (Powers et al., 2006), sequence based methods can be applied to predict the function of a protein. Unfortunately, the other direction does not hold, i.e. proteins with similar functions can exhibit sequences with very low sequence identity (Chalk et al., 2004), like e.g. trypsin and subtilisin (Schmitt et al., 2002). Hence, using sequence based methods for the search of proteins having same function might miss correct hits. Therefore, the overall fold is often considered to predict the function of a protein more reliably (Chalk et al., 2004; Powers et al., 2006), since tertiary structure is evolutionary more conserved (Powers et al., 2006). Following this observation, methods based on the tertiary structure seem more appropriate for function prediction. Nevertheless, one has to assert that sequence- and fold-based methods consider the protein as a whole even though the molecular function of a protein is often determined by its binding site. Therefore protein binding sites are considered instead of whole proteins in this thesis.

1.1 Protein Binding Sites in Drug Design

Protein binding sites are depressions on the surface of a protein which are sometimes deeply buried and totally encapsulated. They can bind endogenous ligands that naturally occur in the cells. Endogenous ligands can play the role of a cofactor or a substrate. In both cases they are modified by the protein but the underlying principle is different. Cofactors contribute a part of the molecule to the catalytic reaction and are regenerated by other enzymes. Substrates are modified by the catalytic reaction to products that are subsequently needed for other reactions, e.g. in metabolic pathways. To manipulate such processes, molecules can be used that are able to bind a certain binding site leading to a suppression or activation of the reaction of that protein. This principle is illustrated in Figure 1.1 where an inhibitor is blocking the protein binding site (right) leading to a suppression of the generation of the two products (left).



(a) Protein binding site bound by a substrate; the reaction leads to a modification of the substrate resulting in two products.

(b) Protein binding site blocked by an inhibitor. A reaction with the substrate is avoided, hence products cannot be generated.

Figure 1.1: Molecule binding a protein: Using inhibitors, chemical processes can be blocked leading to an interruption of whole biological or chemical processes.

One goal in medicinal chemistry is therefore to identify target proteins that are responsible for a disease and to design ligands influencing these target proteins. These ligands should have high potency and selectivity to ensure that only the target proteins are influenced, thus, to avoid possible cross-reactivities. Such cross-reactivities occur if a ligand designed to bind a target also

tends to bind to other proteins leading to undesired effects. With information about protein binding sites such cross-reactivities can be determined by searching for proteins which exhibit binding sites with similar shape and similar distribution of the physicochemical properties. A generally accepted statement is that proteins with similar binding sites can exhibit the same function. Here however, amino acids flanking the protein binding site are not directly considered. In fact, physicochemical properties which define certain interactions between protein binding site and ligand are of much higher interest and are considered instead of amino acids. Hence, to determine the function of a protein again similarity measures are required that can support the development of novel drugs. Drugs should have high potency and selectivity which can be increased by using additional information such as alignments or conserved patterns (see Section 2.5 for an explanation of these terms). Having calculated the similarity between two binding sites, information about their similarities and differences (leading to the obtained overall similarity) can be exploited to develop ligands with maximal selectivity for the target. With methods that process a set of more than two structures, it is even possible to analyze a whole family of proteins, hence to detect conserved patterns which are functionally important. In particular, proteins from different organisms catalyzing the same function often exhibit similar patterns, which are assumed to be responsible for the function of these proteins. Further applications are the binding site based classification of proteins into different families, which use, e.g. a clustering procedure for this purpose. Hence, all these techniques provide valuable information for rational drug design.

Since the number of determined three-dimensional structures is increasing constantly, providing important information for the detection of novel targets, powerful database systems are required allowing the efficient use of this information. In particular, a fast similarity retrieval is of highest importance. With CavBase (Schmitt et al., 2002) a database system was introduced which allows exactly such calculations directly on protein binding sites. This database system will be introduced in the following.

1.2 CavBase

In the previous chapter some problems were presented for which the consideration of sequences or folds may not be optimal. Here, protein binding sites

seem to be the suitable objects of interest. For that reasons, Schmitt et al. (2002) introduced a database system for the automated detection, extraction, storage and comparison of protein binding sites from experimentally determined protein-ligand complexes available through the ReliBase database (Hendlich et al., 2003). In CavBase, labeled points in the 3-dimensional Euclidean space are used as a first approximation to describe binding pockets. The database (release 2.2.2) currently contains 275,162 hypothetical binding pockets that have been extracted from 66,798 publicly available protein structures by using geometric techniques for the cavity detection and a set of rules for the transformation of the amino acid-based representation into a more compact representation based on 3-dimensional descriptors, so-called pseudocenters. For the comparison of the derived cavities, the point representation is further transformed into node-labeled and edge-weighted graphs. Afterwards subgraph-isomorphism-based techniques are applied to retrieve similarity. Figure 1.2 shows the gen-

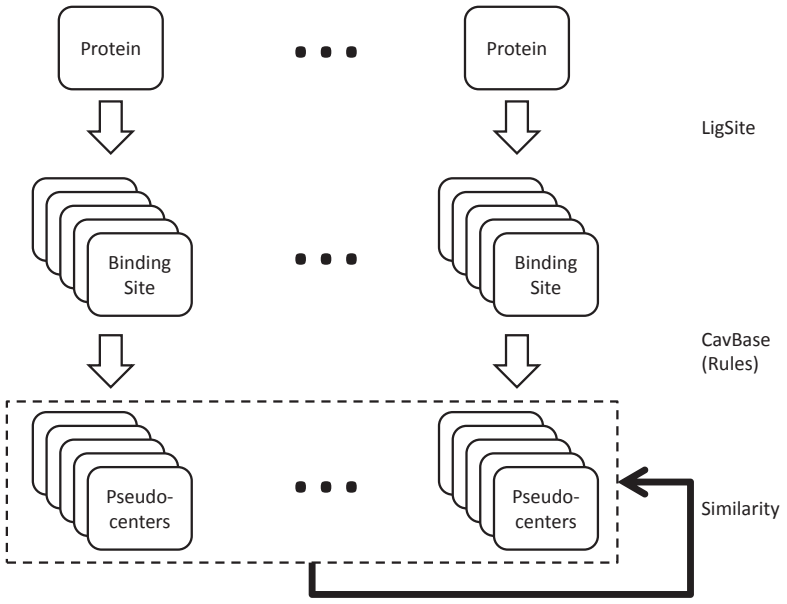


Figure 1.2: Visualization of the CavBase steps: First the LigSite algorithm is used to detect and extract protein binding sites. Subsequently, a set of rules is applied to transform the amino acid representation into a more compact representation in the form of pseudocenters. Protein binding sites are stored in this compact form in CavBase which allows a search for similar protein binding sites by means of subgraph-isomorphism.

eral procedure of CavBase that consists of the three steps: Extraction of binding

pockets, their representation and similarity retrieval. These three steps are described in the following in more detail.

1.2.1 Detection of Protein Binding Sites

For the detection of protein binding sites several methods can be applied, based on different concepts, such as geometry or interaction energy (Pérot et al., 2010). In CavBase, however, the LigSite algorithm (Hendlich et al., 1997) was implemented to detect protein binding sites without using information about bounded ligands by considering its geometry only. According to the authors of the LigSite algorithm, the two main advantages LigSite provides are its independence on the orientation of the protein and that it does not require any (human) knowledge that could bias the approach into a certain, possibly undesired direction.

LigSite uses a regular Cartesian grid in which the protein is embedded, where the grid size is an adjustable parameter. It was recommended by the authors to set this parameter between 0.5 Å and 0.75 Å to find a good trade-off between efficiency and solution quality, where the latter criterion was measured in terms of the smoothness of the surfaces produced. Once the protein is embedded into the grid, those grid points are excluded that are solvent, decided by assessing whether a protein atom is located in 3 Å distance of the considered grid point. For all remaining points integers are used that specify the degree of burial. For a grid-point the integer is determined by considering iteratively the x-, y- and z-axis and the four diagonals appearing in a cube. If along an axis an event appears of the form *protein – solvent – protein* the integer that is set to zero in the beginning is increased by one. Therefore, for each grid-point values between zero and seven are possible. LigSite uses two further parameters min_{int} and a threshold t that are used to influence the creation of the binding site surface. To generate this surface an arbitrary grid-point with associated integer larger than min_{int} is chosen. Those neighbor grid-points also with integer larger than min_{int} are added and the process continues until no neighbor exists fulfilling this property. However, there can still exist points with associated integer above min_{int} that were not yet processed (because they are not connected with the patch). Therefore, a point from this set is considered and the procedure is restarted forming another binding site of the protein. Having identified a binding site, a test is performed as to whether the number of grid points is

below the parameter t, since only such cavities are considered that have a size larger than or equal to t. Finally, all amino acids lying in the distance of 1.1 Å plus *van der Waals radius* to one of the surface points are assigned to the surface.

In this thesis, the cavity-extraction is not considered further, instead the LigSite realization of CavBase is used directly, in which the grid-size parameter was set to 0.5 Å and where the remaining parameters were chosen as $t = 320$ and $min_{int} = 4$.

1.2.2 3-Dimensional Descriptors and Projection onto Surface

CavBase does not consider the atoms of amino acids surrounding the surface of the protein binding site that was extracted by using the LigSite algorithm. This has several reasons, probably the most important being that one is usually interested in a compact representation of the protein binding site. For the comparison of proteins, e.g., many authors (Shatsky et al., 2004; Holm and Park, 2000; Holm and Sander, 1993) use C_α atoms to have a reduced representation of the protein. In the case of protein binding sites such a representation however would come with a loss of information since possible interactions between ligand and protein binding site would not be considered.

Therefore, Schmitt et al. (2002) decided to consider different types of interactions that were extended in a subsequent work (Kuhn et al., 2006) by additional interaction types. In this thesis the complete set of in total seven physicochemical properties is considered, namely *acceptor*, *aliphatic*, *aromatic*, *donor*, *doneptor*, *metal* and *pi*. Each amino acid retrieved by LigSite can possess one or more such physicochemical properties which are extracted by using certain rules given in (Kuhn et al., 2006). This procedure leads to a transformation of the amino acid representation to a representation in the form of a (small) set of pseudocenters.

Subsequently, after extraction of all pseudocenters the physicochemical properties of the pseudocenters are mapped onto the surface points according to the following procedure: Two vectors are assigned to each pseudocenter, namely the vector $\boldsymbol{v}$ that describes the mean orientation along which a certain interaction can be formed. Information about such orientations are determined experimentally and retrieved from the IsoStar (Bruno et al., 1997) database. The vector $\boldsymbol{r}$ is determined as the mean over all vectors, each of which targets from a particular pseudocenter to a surface point that has a distance of at most 3 Å to

that pseudocenter. This procedure is illustrated in Figure 1.3. The angle θ de-

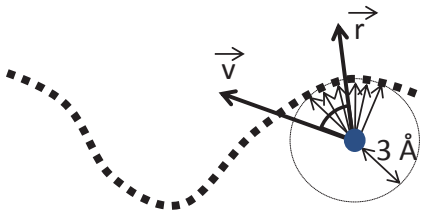


Figure 1.3: Assignment of physicochemical properties onto surface points (due to simplicity illustrated in the 2-dimensional instead of 3-dimensional space): All surface points within radius 3 Å around the pseudocenter (blue point) are considered to generate a surface patch. For each point a vector targeting from the pseudocenter to the point is calculated and finally substituted by the mean-vector $\boldsymbol{r}$. The direction of the vector $\boldsymbol{v}$ is retrieved from the IsoStar data bank. The angle between $\boldsymbol{r}$ and $\boldsymbol{v}$ is specified by θ.

termined is used to decide whether a pseudocenter can assign its property onto the surface. For each physicochemical property, a threshold is defined, where the exact values and the reason for the choice is summarized in (Schmitt et al., 2002; Kuhn et al., 2006). If the angle θ of a pseudocenter exceeds its threshold, the pseudocenter is discarded because it is assumed that the pseudocenter cannot form an interaction with a ligand. The physicochemical properties of the remaining pseudocenters are exposed towards the surface. For a pseudocenter, those surface-contacting grid points are considered, for which the distance to the pseudocenter is below or equal to 3 Å . The physicochemical property of the considered pseudocenter is assigned to all these grid points. Where a property was already assigned to a grid point by another pseudocenter, the property is overwritten if the actual pseudocenter is closer to the grid point than the former one. At the end of this procedure, the grid points describing the surface of the protein binding site are decomposed into patches of certain physicochemical properties.

1.2.3 Similarity Retrieval

The similarity measure in CavBase considers two protein binding sites as similar if they have a similar distribution of the physicochemical properties and a similar geometry. To determine these two properties, CavBase uses the pseudocenters that are a more compact representation of the protein binding site,

thus, that allow more efficient calculations. The similarity calculation in CavBase consists of two steps (Schmitt et al., 2002): In the first step, the 3-dimensional descriptors are transformed into a node-labeled and edge-weighted graph and a search for common subgraphs is performed afterwards, by using clique detection on the product graph (cf. Section 2.1.2). Once these subgraphs are detected, a simple similarity measure can be defined by applying the rule: "The larger the largest common subgraph, the more similar both structures are". The authors, however discarded such an approach, since graphs do not allow distinguishing between concave and convex areas (cf. Figure 1.4) which obviously could lead to undesired results. Hence, all common subgraphs are considered (instead of the largest one), and for each of these subgraphs the second step is performed.

In the second step, for each common subgraph found which defines a partial one-to-one correspondence between the pseudocenters, an optimal superposition is calculated, leading to a transformation rule. Having applied this transformation rule to the whole protein binding site, for pairs of pseudocenters sharing equal label, the overlap is calculated expressed by the number of overlapping surface points (within a distance of 1 Å). Mathematically, this can be expressed by

$$\sigma = \frac{\rho_i + \rho_j}{|S_i| + |S_j|} ,$$

where the sets S_i and S_j, respectively, contain surface points that are assigned to the i-th or j-th pseudocenter. The variable ρ_i gives the number of surface points represented by the i-th pseudocenter that are located within a 1.0 Å distance to at least one surface point represented by the j-th pseudocenter; ρ_j is defined analogously. The authors in (Schmitt et al., 2002) moreover wanted to avoid the consideration of strongly fragmented surface patches, therefore the final score was calculated not as sum over σ but instead by

$$S = \sum_{\sigma \geq 0.7} \sigma ,$$

thus only those sigma values were considered for which the mutual overlap was above 70%.

In a final refinement step, a new transformation is computed, now based on the subset of three-dimensional descriptors that passed the 70% condition. Again, this transformation is applied to the whole protein binding site and the similarity S is recalculated. This recalculated value S is used in CavBase to

express the similarity between two protein binding sites. Another used value is determined by

$$S' = \frac{S - 0.7n}{rmsd} ,$$

where n is the number of nodes in the currently considered common subgraph, and where *rmsd* is the root mean squared deviation (cf. Definition 2.4) of the transformation calculated in the last step of this procedure. As the mutual overlap must exceed 70% of the considered points, the variable S ranges in the interval $[0.7 \cdot n, n]$, where n is the number of surface points. Therefore, the numerator takes values between 0 and $0.3 \cdot n$, where larger values correspond to stronger similarity. The *rmsd* takes per definition positive values that grow with increasing deviation. Hence, with S' again a similarity measure is obtained, that is, as the authors in (Schmitt et al., 2002) claim, more robust against fragmented and disconnected motifs.

At the end of this procedure, a set of similarity values $\{S\}$ and $\{S'\}$ is calculated of which the pair is taken that was determined by the common subgraph which leads to the highest S value. Schmitt et al. (2002) proposed to consider only the 100 largest common subgraphs, which seems to be a good trade-off between efficiency and quality of the solution; moreover, the authors proposed using S for similarity retrieval.

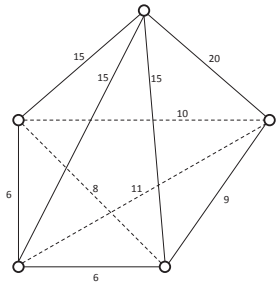
1.3 Goals of this Thesis

Considering CavBase and its architecture (cf. Figure 1.2), there are different possibilities for improvements. Kuhn (2004) considered for example the second level of Figure 1.2 and developed another representation of protein binding sites, in which a vector was used which specifies to which degree the seven physicochemical properties are present for a pseudocenter. The first level of CavBase was investigated by different researchers where different concepts were proposed for the detection of protein binding sites based on different algorithmic and biological concepts (Pérot et al., 2010).

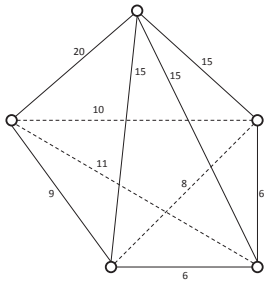
In this thesis, similarity retrieval, hence the last level of Figure 1.2, is considered, which is obviously the bottle-neck of CavBase. In difference to the other levels this step is applied several times for a protein binding site, leading to the requirement of a very efficient realization. However, searching for common subgraphs in terms of a clique detection on the product graph comes with some

drawbacks: The product graph of two graphs $G = (V, E)$ and $G' = (V', E')$ consists of $\mathcal{O}(|V| \cdot |V'|)$ nodes, hence of $\mathcal{O}(|V|^2 \cdot |V'|^2)$ edges, which becomes too high a number even for middle-size graphs. Thus, such approaches often cannot be applied for the comparison of protein binding sites. Even if the product graph can be placed in the memory, the identification of cliques in a graph is known to be NP-hard, hence all algorithms solving the problem are a compromise between efficiency and quality of the solution. CavBase realizes the clique detection by the (Bron and Kerbosch, 1973) algorithm that is exact, yet exponential in runtime (Tomita et al., 2006). However, usually the time needed to detect the cliques does not dominate the whole approach. In fact the calculation of the set of values $\{S\}$ derived from the cliques is the most expensive part of the similarity retrieval in CavBase, since the expensive scoring must be performed 100 times.

The goal of this work is therefore to enable on the one hand more efficient measures between protein binding sites. Like in CavBase, this problem can be considered on the level of graphs, the representation used so far for calculations on protein binding sites. However, already the authors of CavBase reported that this representation is not optimal, which leads to the additional steps in CavBase taking the surface points into consideration. Therefore, on the other hand a more robust representation is considered which is based on labeled point clouds. CavBase stores protein binding sites obviously as such labeled point clouds in the 3-dimensional Euclidean space that can be used di-



(a) graph representing a structure A



(b) graph representing a structure B different from A

Figure 1.4: Example for two equivalent graph descriptors (circles and lines) describing geometrically different structures (circles).

rectly for calculations. In comparison to graphs, this representation does not come with a loss of information effective when distances are considered instead

of coordinates, as demonstrated in Figure 1.4.

A widely used technique to measure similarity between objects is to use a vector representation on single objects and measures on pairs of vectors. Therefore, in addition to the graph-based representation and the novel geometric representation of protein binding sites discussed until now, vector representations are also developed. Since entries in the vector are called features, such approaches are also referred to as feature-based approaches. Even though the transformation from a protein binding site to a vector is usually more complex compared to the transformation into a graph, this representation has the benefit that the comparison of vectors applied afterwards is much more efficient than a comparison of graphs. Since the transformation must be applied once for each protein binding site whereas comparisons are performed multiple times, a vector representation which is stored in parallel to the protein binding site will lead to a enormous gain of efficiency.

Independent of the issue of representation, more advanced techniques are required to handle noise and mutation often appearing in life-sciences. In the case of protein binding sites, error-tolerance is moreover required to handle different conformations caused by an *induced fit*. In CavBase some error-tolerance was enabled by using an appropriate definition of the product graph, where edges are assumed equivalent within a predefined threshold of ϵ. Moreover, with the concept of subgraph-isomorphism, an algorithmic error-tolerance was allowed, since similarity can be obtained even if the graphs do not match exactly. The error-tolerance provided by these concepts, however, is quite low, which necessitates the development of more robust methods and the investigation of their behavior in comparison to less flexible methods.

Finally, motivated by the work of Weskamp et al. (2007), the problem of multiple structural alignments is also considered in this thesis. This problem has so far been tackled in a similar way to the calculation of similarity in CavBase. The common subgraph is calculated that obviously forms a partial alignment which is greedily extended to a complete alignment. Using star-alignment (Böckenhauer and Bongartz, 2007), the state-of-the art technique for merging pairwise alignments to multiple ones, a multiple alignment is formed. This greedy procedure does not perform optimally in most cases, hence other techniques should be applied to calculate such a graph-alignment. Moreover, motivated by the drawbacks graphs exhibit, an analogous approach is required for the geometric representation of a protein binding site.

2

Preliminaries

In the previous chapter important problems pharmaceutical chemists have to tackle were discussed, namely the extraction of similarity and the construction of alignments. Before presenting algorithms capable of solving these tasks, in this chapter an introduction to fundamental tools is given, which are used in the following of this thesis. Here the discussion comprises techniques allowing the modeling of protein binding sites, but also techniques that allow for optimizing complex functions for which gradient-based approaches will fail or that allow the processing of imprecise and vague information.

For modeling protein binding sites at least two approaches can be applied: As already done in CavBase, a model based on graphs can be used. Another method introduced in this thesis adopts a novel representation based on points in the 3-dimensional Euclidean space, which offers some advantages compared to graphs. To solve optimization problems that are in this thesis often endowed with properties making the optimization hard, evolutionary algorithms become an important tool. Here, evolutionary strategies turn out to be especially useful. Another useful concept is provided by fuzzy logic, offering a large set of tools to model and process vague and imprecise data.

Parts of this chapter were already discussed and published in (Fober et al., 2007), (Fober et al., 2009c), (Fober et al., 2009d), (Fober et al., 2011) and (Hüllermeier et al., 2013).

2.1 Modeling of Protein Binding Sites

In Section 1.2, the database *CavBase* was introduced as a tool for the automated detection, extraction, and storing of protein cavities from experimentally de-

termined protein structures available through the *PDB*. In CavBase, protein binding sites are stored in an already compact representation. This representation considers pseudocenters, spatial points in the Euclidean space that are labeled with one of seven physicochemical properties. For storing binding sites, CavBase uses a simple list in which each row represents a pseudocenter by its coordinates in the Euclidean space and an associated physicochemical property. Thus, data are given as a set of points in the Euclidean space that are labeled with elements from a discrete set. In this thesis, such data is called *labeled point cloud*.

Instead of considering coordinates in the Euclidean space, Havel et al. (1983) proposed considering distances which eventually lead to a representation of protein binding sites in the form of graphs. On the one hand, this representation comes with the advantage that an enormous number of methods exists to process these graphs. On the other hand, a graph representation comes with the drawback of a larger memory consumption since there are $\binom{n}{2}$ distances in a labeled point cloud of size n, and furthermore, with the problem that the original point cloud cannot be reconstructed from a given graph representation. This leads to an inevitable loss of information in the general case, as illustrated in Figure 1.4.

2.1.1 Point Clouds

To model geometrical objects, sets of points, so-called point clouds, are often used. This concept is very primitive and considers a set of points expressed in the form of coordinates, usually in the 3-dimensional Euclidean space. Generally, point clouds are not as universal as graphs, however, they allow for modeling geometric data of a certain dimension and this in a very efficient way. Hence, for various data like manufactured parts (Thompson et al., 1999), volumetric data (Fabio, 2003), geographic information (Höfle et al., 2007) and many others, such a representation is used, since all important information about the external surface of an object can be captured in an efficient way.

As mentioned, protein binding sites are represented by points in the 3-dimensional Euclidean space, which are moreover enriched with a physicochemical property. Therefore, it is not sufficient to consider (unlabeled) point clouds since important information could not be represented and hence not processed. Instead, in this thesis labeled point clouds are introduced that ex-

tend the original point cloud by labels. A labeled point cloud is a finite set of points, where each point is not only associated with a position in n-dimensional Euclidean space, but also with a discrete class label that can represent a certain attribute. Hence, a labeled point cloud P of cardinality m is given by

$$P = \{(x_1, \ell_1), (x_2, \ell_2), \ldots, (x_m, \ell_m)\} \subset \mathbb{R}^n \times \mathcal{L} ,$$

where $x \in \mathbb{R}^n$ and $\ell \in \mathcal{L}$ is the label of a point x. For modeling protein binding sites, it is obviously sufficient to consider points in the 3-dimensional Euclidean space, whose labels are used to represent the physicochemical properties. The entries in CavBase are obviously given exactly in the form of labeled point clouds. Hence, for algorithms working on those labeled point clouds, the raw data can be directly processed without the need for a certain transformation that often is afflicted with a loss of information.

Important measures on point clouds are given by the Euclidean distance, the one-sided Hausdorff distance and the Hausdorff distance, which is basically a combination of the one-sided distances. In bioinformatics, the root mean square deviation is a further often applied measure on point clouds.

The Euclidean distance is used to measure the distance between two points p and p' in an n-dimensional space.

Definition 2.1 (Distance based on norms)
The norm-based distance between two pints $p, p' \in \mathbb{R}^n$ is defined as

$$d(p, p') = \left(\sum_{i=1}^{n} \left| p_i - p'_i \right|^k \right)^{\frac{1}{k}} .$$

Special realizations of this distance are the Manhattan (L_1) distance d_{MH} ($k = 1$), the Euclidean distance d_E ($k = 2$) and the infinity-norm distance d_∞ ($k = \infty$). The one-sided Hausdorff distance, that makes use of the Euclidean distance, measures the distance from point cloud P to point cloud P'.

Definition 2.2 (One-sided Hausdorff distance)
For $P, P' \subset \mathbb{R}^n$

$$\delta_{\rightarrow}(P, P') = \max_{p_i \in P} \min_{p'_j \in P'} d_E(p_i, p'_j) .$$

is called one-sided Hausdorff distance.

To obtain finally the distance between two point clouds P and P', hence their Hausdorff distance, two one-sided Hausdorff distances, namely $\delta_{\rightarrow}(P, P')$ and $\delta_{\rightarrow}(P', P)$, are combined, resulting in the following definition:

Definition 2.3 (Hausdorff distance)
For $P, P' \subset \mathbb{R}^n$ the Hausdorff distance is defined as

$$\delta(P, P') = \max\{\delta_{\rightarrow}(P, P'), \delta_{\rightarrow}(P', P)\} .$$

Obviously, the Hausdorff distance in its original form is not able to process label information, moreover, coordinates in the Euclidean space are dependent on the origin of the coordinate system. This makes a comparison of objects that do not share the same origin in the coordinate system more difficult, since a common origin must be found in a first step.

Another measure on point clouds for which a one-to-one correspondence between points exist is their *root mean squared deviation (rmsd)*, which expresses the quality of the superposition derived from the one-to-one correspondences in terms of the (Kabsch, 1976) algorithm. Given optimal correspondences, the superposition is the better, the more similar the point clouds are. Hence, the rmsd can be considered as a further similarity measure on protein binding sites and can be calculated as follows:

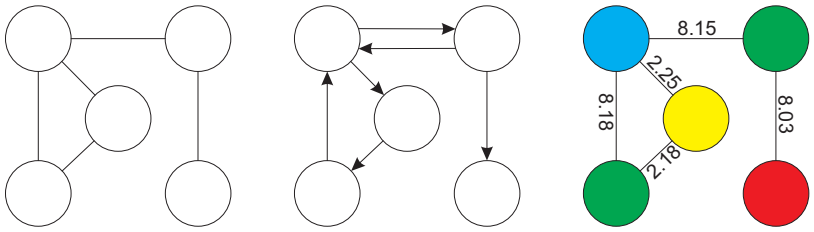
Definition 2.4 (Root mean square deviation (rmsd))
Let P and P' be two point clouds of size n which were superimposed optimally. Moreover, let the indices of the points in P and P' represent the one-to-one correspondences, i.e., $p_i \in P$ and $p'_i \in P'$ correspond to each other. Then the rmsd is defined as

$$\mathrm{rmsd}(P, P') = \sqrt{\sum_{i=1}^{n} \frac{(p_i - p'_i)^2}{n}} .$$

2.1.2 Graphs

Graphs are very flexible and powerful tools for modeling and representing various data already long in use in chemo- and bioinformatics. A graph $G = (V, E)$ consists of a non-empty set V of nodes and a set $E \subseteq V \times V$ of edges. An edge $e = (v_i, v_j) \in E$ connects two nodes, therefore it puts two nodes in a (binary) relation. In mathematics one can consider symmetric or asymmetric relations leading to undirected or directed graphs, respectively. For undirected graphs, hence graphs for which $(v_i, v_j) \in E \Leftrightarrow (v_j, v_i) \in E$, it would be more correct to use a subset instead of a tuple representation. For convenience, however, the simpler tuple notation is used here, with the implicit understanding that $(v_i, v_j) \in E$ implies $(v_j, v_i) \in E$.

To increase the expressiveness of a graph, one can label its nodes and edges with two sets of node and edge labels $\mathcal{L}_V$ and $\mathcal{L}_E$. To this end, a graph G will be extended by two functions $\ell_V : V \to \mathcal{L}_V$ and $\ell_E : E \to \mathcal{L}_E$ that assign labels from these sets to nodes and edges, respectively. This leads to a graph $G = (V, E, \ell_V, \ell_E)$. Often $\mathcal{L}_V$ is a discrete set and $\mathcal{L}_E$ is the (real) numbers. In this case, G is called a node-labeled and edge-weighted graph.



(a) undirected graph

(b) directed graph; direction by arrow

(c) undirected and labeled graph; edge weights by numbers and node labels by colors

Figure 2.1: Example for undirected, directed and undirected labeled graphs.

In Figure 2.1, different types of graphs are illustrated, each of which is appropriate for a certain application. While undirected and unlabeled graphs can be used to model relations such as constraint networks, for flow or scheduling problems directed graphs should be used. For modeling geometric or chemical objects, undirected, node-labeled and edge-weighted graphs are an appropriate and widely used representation. This representation will be considered in the following for the special case of geometric graphs, where nodes represent elements in the Euclidean space that however have lost their coordinates. Instead, the edge weights are used to capture geometric information in the form of the Euclidean distance between pairs of nodes. In the following, some definitions that will be used in this thesis are recalled for undirected node-labeled and edge-weighted graphs.

Definition 2.5 (Size of a graph)

The size of a graph G is defined as the number of nodes appearing in G, thus $\text{size}(G) = |V|$. *The number of edges in a graph of size n can be bounded by* $\binom{n}{2} = \mathcal{O}(n^2)$.

An important identification number of a graph are the degrees of its nodes. The degrees of the nodes can be used e.g. to determine whether a graph is

connected or complete. The degree of a node is defined as follows:

Definition 2.6 (Degree of a node)
For a node $v_i \in V$ of an undirected graph $G = (V, E)$, the degree *of v_i is defined as the number of edges that are connected with v_i. Formally, the degree of a node v_i in a graph G is given by*

$$\deg_G(v_i) = \left| \{ v_j \mid (v_i, v_j) \in E \} \right| .$$

To traverse a graph, different concepts can be used, namely walks and paths.

Definition 2.7 (Walk)
A sequence of nodes $(v_1, \ldots, v_n)$ is called walk *of length n, if and only if $(v_i, v_{i+1}) \in E$ for all $i = \{1, \ldots, n-1\}$. For labeled graphs, a walk emits a sequence either of node labels $\big(\ell_V(v_1), \ldots, \ell_V(v_n)\big)$, edge weights $\big(\ell_E(v_1, v_2), \ldots, \ell_E(v_{n-1}, v_n)\big)$, or a combination of both $\big(\ell_V(v_1), \ell_E(v_1, v_2), \ldots, \ell_E(v_{n-1}, v_n), \ell_V(v_n)\big)$.*

In a walk, nodes can be traversed several times, thus the number of walks and the length of a walk, specified by the number of nodes, cannot be bounded, and hence can become theoretically infinite. To reduce the number of walks, the concept of a path can be used.

Definition 2.8 (Path, cycle)
A path is a walk in which a node appears no more than once. Another but equal definition is based on cycles, thus on a walk $(v_1, \ldots, v_n)$ in which $v_n = v_1$, and a path is defined as a walk without any cycle.

The maximal length of a path in a graph of size n is $\mathcal{O}(n)$. However, the number of paths in a graph is still exponential in the number of nodes, thus it becomes high even for small graphs. To reach a further reduction, one can consider the shortest paths whose number is $\mathcal{O}(n^2)$ in a graph of size n. To find *all* shortest paths in a graph, a couple of algorithms can be used (Corman et al., 2001), most of which use dynamic programming techniques and come with a complexity of $\mathcal{O}(n^3)$.

Definition 2.9 (Connected graph)
A graph is called connected if every pair of distinct nodes u and v is connected, thus if there is a path from u to v. Non-connected graphs become decomposed into connected components that are maximal connected graphs.

To model geometric structures, it is obviously necessary to work on connected graphs, otherwise the geometry between the disconnected parts would become

completely unrecoverable. Moreover, all nodes must have a certain degree to ensure a certain degree of rigidity. The term rigidity becomes important if considering geometric graphs, which are graphs representing a geometric object. Obviously, the less connected the nodes are, the less information about the geometry is available, hence the more geometries can be represented by the same graph. An example for this phenomenon is a graph in which one node is connected by only one edge with the rest of the graph. Even though the edge weight is fixed, there are still degrees of flexibility (e.g. torsion-angles) which lead to the same graph for different geometries. Hence, by varying the number of edges, the degree of error-tolerance can be influenced, too. The most rigid graphs in this context are so-called cliques.

Definition 2.10 (Complete graph, clique)
A graph is complete if every pair of distinct nodes is connected by an edge. Complete graphs are also called cliques. To test if a graph is a clique, the degree can be used. A graph of size n in which all nodes have degree $(n-1)$ *is obviously a clique. Another approach is to count the number of edges that must be* $\binom{n}{2}$ *in a clique.*

Beside a win of flexibility, incomplete graphs lead for many algorithms to a win of efficiency, since the number of edges is often part of the complexity estimation. Two important terms appearing in combination with cliques are *maximum* and *maximal*: Where the maximum clique in a graph is the largest subset of nodes that form a clique, the maximal clique is a subset of nodes that form a clique which cannot be extended by further nodes. The differences are illustrated in Figure 2.2, where the solid circle is used to mark the maximum clique. However, beside the maximum clique, there appear two further maximal cliques (note that the maximum clique is also always a maximal clique) marked with dashed circles. These cliques cannot be extended by further nodes without violating the clique property.

Many concepts exist for representing and storing graphs. The most common approaches are adjacency matrices, adjacency lists and incidence lists. The adjacency matrix is defined as follows:

Definition 2.11 (Adjacency matrix)
Let G be a graph of size n. Then the adjacency matrix is given by an $n \times n$ *matrix A, where*

$$[A]_{i,j} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}.$$

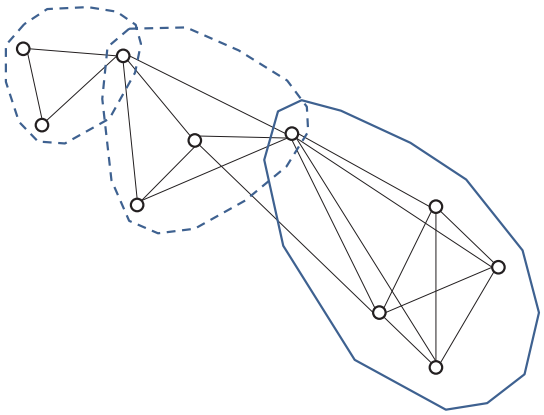


Figure 2.2: Difference between the maximum clique (solid circle) and maximal cliques (solid and dashed circles).

For node-labeled and edge-weighted graphs without cycles of length 1*, as used here, a matrix representation can be defined as*

$$[A]_{i,j} = \begin{cases} \ell_E\big((v_i, v_j)\big) & \text{if } i \neq j \text{ and } (v_i, v_j) \in E \\ \infty & \text{if } i \neq j \text{ and } (v_i, v_j) \notin E \\ \ell_V(v_i) & \text{if } i = j \end{cases} .$$

In case of sparse graphs as well as undirected graphs, such a representation of course will lead to quite inefficient storage. However, a matrix representation will allow one to operate on the representation directly, thus leading to a gain of efficiency in terms of runtime.

More space efficient representations are *adjacency lists* (Corman et al., 2001), which can be considered as a compressed representation of a sparse adjacency matrix. The adjacency list contains pairs of nodes that are connected by an edge, hence that are adjacent. As an alternative, the upper (or lower) triangle-matrix can also be considered, which would decrease the memory requirement by a factor of two. Incidence matrices or lists (Corman et al., 2001) set nodes and edges in relation. Incidence matrices are always sparse, since each edge contains exactly two nodes.

For testing two graphs for equivalence, concepts based on graph isomorphism are often used, defining equivalence through isomorphism.

Definition 2.12 (Graph isomorphism)

For two graphs $G = (V, E, \ell_V, \ell_E)$ *and* $G = (V', E', \ell'_V, \ell'_E)$*, a bijective function*

$f : V \to V'$ is called graph isomorphism if it satisfies the property $(u,v) \in E \Leftrightarrow (f(u), f(v)) \in E'$. For the labeled case $\ell_V(v) = \ell'_V(f(v))$ must hold and $(u,v) \in E \Leftrightarrow (f(u), f(v)) \in E'$ is substituted by $\ell_E\Big((u,v)\Big) = \ell'_E\Big((f(u), f(v))\Big)$.

If for two graphs G and G' there exists such a function f, G and G' are called isomorphic, symbolized $G \approx G'$. Often, one is not interested in the whole graph but instead in important parts of the graph, hence in a subgraph of a graph.

Definition 2.13 (Subgraph)
A graph $G' = (V', E')$ is called a subgraph *of $G = (V, E)$, if and only if $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$. It is called an* induced subgraph, *if and only if $V' \subseteq V$ and $E' = E \cap (V' \times V')$.*

Special types of subgraphs are *neighborhood graphs* and *maximum common subgraphs* that are defined in the following.

Definition 2.14 (Neighborhood graph)
For a graph $G = (V, E)$ and a node $v \in V$, the graph $G^{(v)} = (V_v, E_v)$ is called a neighborhood graph of v, if $V_v = \{v\} \cup \{v_i \in V \mid (v, v_i) \in E\}$ and $E_v = E \cap V_v^2$.

Definition 2.15 (Maximum common subgraph)
A maximum common subgraph of two graphs G and G' is the largest graph G_{mcs} that is a subgraph of G and G' as well. In the literature the terms maximum common induced subgraph (mcis) and maximum common edge subgraph (mces) are distinguished. While the mcis has the maximal number of nodes, algorithms solving the mces problem look for the maximal number of edges. Here the maximum common induced subgraph is used and will subsequently be called maximum common subgraph (mcs).

The concept of a supergraph is inverse to that of a subgraph: A graph G' is called a supergraph of another graph G, if G is a subgraph of G'.

An important binary operation on graphs is its product, that is a mapping $\mathcal{G} \times \mathcal{G} \to \mathcal{G}$ which has many realizations. The three most common are the *Cartesian*, the *categorical* and the *strong product graph* (Harary, 1994). Due to its important properties, the categorical product graph will be used in this thesis.

Definition 2.16 (Categorical product graph)
Given two graphs $G = (V, E, \ell_V, \ell_E)$ and $G' = (V', E', \ell'_V, \ell'_E)$ and a threshold for

edge matches ϵ, the categorical product graph is defined as $G_\times = (V_\times, E_\times)$, where

$$V_\times = \left\{ (v_i, v'_j) \in V \times V' \mid \ell_V(v_i) = \ell'_V(v'_j) \right\}$$

$$E_\times = \left\{ ((v_i, v'_j), (v_k, v'_l)) \in V_\times^2 \mid \|\ell_E(v_i, v_k) - \ell'_E(v'_j, v'_l)\| \leq \epsilon \right\}$$

in the case of node-labeled and edge-weighted graphs.

The product graph $G_\times = (V_\times, E_\times)$ of G and G' has a number of interesting properties that allow one to discover patterns such as common substructures or common walks in two graphs. A walk $(v_1, \ldots, v_n)$ in the product graph corresponds to a common walk in G and G'. In the graph G, this walk starts in node $[v_1]_1$ and ends in node $[v_n]_1$. Correspondingly, it starts in node $[v_1]_2$ and ends in node $[v_n]_2$ in G'. The product graph can also be used to calculate the maximum common subgraph of G and G'. The set of nodes of the maximum clique $G_C = (V_C, E_C)$ in the product graph, where $V_C \subseteq V_\times$, corresponds to nodes that appear in G and G' as well (Levi, 1973). Thus, the maximum common subgraph in G is given by the set of nodes $V_{mcs} = \{[v]_1 \mid v \in V_C\}$ and the set of edges $E_{mcs} = V_{mcs}^2 \cap E$, correspondingly the maximum common subgraph in G' is given by $V'_{mcs} = \{[v]_2 \mid v \in V_C\}$ and $E'_{mcs} = V_{mcs}^{'2} \cap E'$. Therefore, to detect common subgraphs, one can simply search for cliques in the product graph $G_\times$, and finding a maximum common subgraph amounts to finding a maximum clique in $G_\times$. In other words, the problem of finding a maximum common subgraph can be reduced to the problem of clique detection, and any algorithm for the latter can be used to solve the former problem. In this regard, it is worth mentioning that clique detection is an NP-hard problem (Karp, 1972). Hence, exact algorithms are feasible only for very small graphs, while practically relevant problems are usually solved in an approximate way by means of heuristic algorithms. A further problem of the product graph is its high space and time complexity. For the construction, first a set of nodes is generated that has cardinality $\mathcal{O}(n^2)$, where $n = \max\{|V|, |V'|\}$, which are afterwards connected by a set of $\mathcal{O}(n^4)$ edges, resulting in a corresponding space and time complexity.

In contrast to graphs, the formerly introduced point clouds have the property that they represent an object in a certain dimension which cannot be changed afterwards. Hence, the number of degrees of freedom is much smaller compared to graphs. Moving, e.g., one point in a point cloud of size n leads immediately to a change of n distances. Hence, the number of degrees of free-

dom is given at most by n degrees for the coordinates and moreover n degrees for the labels. On the other hand, for graphs there are also n possibilities to change the node labels. However, since there are $\binom{n}{2}$ edge weights in a graph of size n, each of which can be modified, the number of degrees of freedom quickly becomes very large. This leads to a very flexible model, however, the consideration of so many degrees of freedom might lead to algorithmic problems and an increased complexity. Moreover it is still an open question if such a flexible model is beneficial for modeling protein binding sites. E.g., such a model allows one to consider a graph of size 3 exhibiting the side length $(1, 1, 9)$. However, such a graph does not represent a 3-dimensional geometric structure, hence it cannot represent a protein binding site. This might lead to an artificial expansion of the search space.

Deriving Graphs from Geometric Data

Obviously, graphs can be used to model various data. Protein binding sites are given in the form of a set of pseudocenters in the Euclidean space. Hence, to model a protein binding site in terms of a graph, the natural way is to represent each pseudocenter of the protein binding site by a node in the graph. Thus a set V is created with associated function $\ell_V : V \to \mathcal{L}_V$, where $\mathcal{L}_V$ is a discrete set containing the pyhsicochemical properties. Although coordinates could be used as additional node labels, graphs are not the best choice to process on these coordinates, since calculations would become difficult. Instead, to capture the geometry, distances between pairs of nodes are considered. For this, a complete graph is constructed by connecting all pairs of nodes $v, w \in V$ by an edge. Additionally, the function $\ell_E : E \to \mathbb{R}_+$ is used to assign to each edge (v, w) a weight that is given by the Euclidean distance between the pseudocenters represented by the nodes v and w.

Finally, a complete node-labeled and edge-weighted graph is generated that captures the physicochemical properties of a protein binding site by node labels and its geometry by edge weights. As already mentioned at the beginning, a drawback of this model is its number of distances which is quadratical in the number of pseudocenters. A clear advantage of this model is that the resulting graph becomes invariant to translation and rotation what can simplify calculations. Moreover, a reduction of the number of edges can be performed. To reduce this number, Weskamp et al. (2007) consider graphs which are not

necessarily complete. The graphs are constructed as described above, however, in a post-processing step, edges whose weights exceed a certain threshold δ are removed. This has two main advantages: On the one hand, the graph becomes more flexible. As already mentioned, complete graphs are the most rigid graphs, hence a removal of edges leads to more (structural) flexibility, thus to a higher error-tolerance that could become advantageous for noisy data such as protein binding sites. On the other hand, the number of edges is reduced, eventually dramatically, leading to an increase of efficiency as many authors claim. Unfortunately, this is not the complete truth. Since non-existing edges in G and G' are represented in the product graph by an edge, the cardinality of the set $E_\times$ is growing with a decreasing number of edges in the input graphs, leading often to higher runtimes of algorithms processing on the product graph.

2.2 Fuzzy Logic

The term *fuzzy logic* is used with different meanings in literature. In a narrow sense, it refers to a branch of mathematical logic, where it is studied as a special type of multivalued logic, i.e., a logic with more than two truth degrees (Hajek, 1998). In a wider (and more common) sense, fuzzy logic is used as an umbrella term for a collection of methods, tools and techniques for constructing intelligent systems that, by virtue of the very idea of the partiality of truth, are capable of handling, processing and exploiting uncertain, imprecise, and incomplete information. These methods build on the key concept of a fuzzy set, as introduced by the founder of fuzzy logic, Lotfi A. Zadeh, in his seminal paper (Zadeh 1965). Fuzzy sets formalize the idea of graded class membership, according to which an element can partially belong to a set. In conjunction with generalized logical (set-theoretical) operators and derived notions like a fuzzy relation, the concept of a fuzzy set can be developed into a generalized set theory, which in turn provides the basis for generalizing theories in different branches of (pure and applied) mathematics as well as fuzzy set-based approaches to intelligent systems design, encompassing methods for information processing, decision making, optimization, and data analysis.

The notion of truth is commonly considered as a bivalent concept: Logically, a proposition is either true or false, but nothing in-between. This conception, which pervades modern science and thinking, has a longstanding tradition in Western philosophy, and manifests itself in standard mathematical

theories, notably logic and set theory. Admittedly, formal systems based on bivalent logic (including theories of uncertainty based on such systems, such as probability theory) have proved extremely useful in the scientific terrain, where they paved the way for the amazing success of the exact and engineering sciences in the last century. In many other, less exact fields of science, however, ranging from the biological and life sciences over legal practice to the modeling of cognitive processes and human intelligence, the bivalence of truth can be called into question. In fact, it was already noticed by Bertrand Russell in 1923 that "all traditional logic habitually assumes that precise symbols are being employed. It is therefore not applicable to this terrestrial life, but only to an imagined celestial existence" (Russell, 1923). Roughly speaking, this is because of the vagueness and ambivalence of the concepts dealt with in these fields: For the intension of these concepts, there is rarely a precise extension in the sense of a set of real objects belonging to that concept in the real word. For example, what is a "short DNA molecule"? Biologists have a vague though sufficiently clear idea of this concept, without using a precise definition in terms of an exact upper bound on the number of base pairs (bp) or the length in μm. Given such bounds, a proposition like "DNA molecule X is small" would be either true or false. Needless to say, this way of adapting human thinking to conventional logic and set theory would be neither desirable nor useful. Instead, fuzzy logic offers an approximation in the other direction: Logic and set theory are generalized, so as so enable a more faithful mathematical modeling of human conception. The core idea in this regard is the notion of a fuzzy set, which allows for partial membership and soft class boundaries (Pedrycz and Gomide, 2007).

Fuzzy Sets

A fuzzy subset A of a reference set X is identified by a so-called membership function, often denoted μ_A which is a generalization of the characteristic function $\mathbb{I}_A$ of an ordinary set $A \subset X$. For each element $x \in X$, this function specifies the degree of membership of x in the fuzzy set; it can be interpreted as the truth degree of the proposition that $x \in A$. Usually, membership degrees $\mu_A(x)$ are taken from the unit interval $[0,1]$, i.e., a membership function is an $X \to [0,1]$ mapping. In principle, however, more general membership scales (such as ordinal scales or complete lattices) can be used. Let $F(X)$ denote the set

of all fuzzy subsets of X. Fuzzy sets are often used for discretizing numerical attributes in a "soft" manner, taking advantage of their ability to model "non-sharp" boundaries between classes. Thus, they serve as an interface between the original numerical scale and a symbolic scale comprised of the (natural language) terms associated with the fuzzy sets. For example, in gene expression analysis, one typically distinguishes between normally expressed, under-expressed and over-expressed genes. This classification is made on the basis of the expression level of the gene (a normalized numerical value), by using corresponding thresholds. For example, a gene is often called over-expressed if its expression level is at least twofold increased. Needless to say, a precise threshold of that kind is arbitrary to some extent and implies an unnatural sudden jump from completely overexpressed to not at all over-expressed.

To operate with fuzzy sets in a formal way, fuzzy set theory offers generalized set-theoretical logical connectives (like in the classical case, there is a close correspondence between set theory and logic). Especially important in this regard is a class of operators called triangular norms or t-norms for short (Klement et al., 2002). A t-norm $\top$ is a $[0,1] \times [0,1] \to [0,1]$ mapping which is associative, commutative, monotone increasing (in both arguments) and which satisfies the boundary conditions $\top(x,0) = 0$ and $\top(x,1) = x$ for all $x \in [0,1]$. A t-norm naturally qualifies as a generalized logical conjunction. Moreover, it can be used to define the intersection of fuzzy subsets $A, B \in F(X)$ as follows: $\mu_{A\cap B}(x) = \top(\mu_A(x), \mu_B(x))$ for all $x \in X$. The logical disjunction can be generalized analogously, namely by means of a t-conorm $\perp$. If $\top$ is a t-norm, then $\perp$ defined by $\perp(x,y) = 1 - \top(1-x, 1-y)$ is a t-conorm. A t-conorm can be used for defining the union of fuzzy sets: $\mu_{A\cup B}(x) = \perp(\mu_A(x), \mu_B(x))$ for all $x \in X$. A generalized implication inc is a $[0,1] \times [0,1] \to [0,1]$ mapping which is monotone decreasing in the first and monotone increasing in the second argument, and which satisfies the boundary conditions $\mathrm{inc}(x,1) = 1$, $\mathrm{inc}(0,x) = 1$ and $\mathrm{inc}(1,x) = x$. In this thesis, the following instances will be considered:

$$\top(x,y) \stackrel{\mathrm{df}}{=} \min(x,y) \ ,$$

$$\perp(x,y) \stackrel{\mathrm{df}}{=} \max(x,y) \ \text{ and}$$

$$\mathrm{inc}(x,y) \stackrel{\mathrm{df}}{=} \max\big((1-x), y\big) \ .$$

2.3 Evolutionary Algorithms

Evolutionary algorithms are powerful optimization algorithms that are based on principles of biological evolution. This is the reason why they differ from classical optimization techniques. Evolutionary algorithms usually do not use gradient information, therefore they belong to the class of direct search methods that require the evaluation of points in the search space only. On the one hand, this makes it possible to apply them on a wide variety of optimization problems, on the other hand, they are not competitive with classical optimization methods on well behaved optimization problems, i.e. optimization problems that are continuously differentiable. While gradient-based approaches are usually processing on one point, evolutionary algorithms use a population of solutions what comes with two main advantages: Evolutionary algorithms can be parallelized in an easy way and have a good performance on multimodal optimization problems. Of course, they become more inefficient in comparison to gradient-based approaches or specialized methods developed for certain problems. However, in this thesis problems will be considered for which it is known that no efficient method exists to solve them due to their NP-hardness and a missing continuous differentiability. For these problems, evolutionary algorithms can be employed that are now introduced in more detail.

Evolutionary algorithms are iterative algorithms that work on a population that contains potential solutions of the optimization problem. Such solutions are called individuals and a population can contain a larger number of individuals specified by an exogenous parameter μ. In the beginning of the evolutionary algorithm, the population is initialized and a loop as illustrated in Figure 2.1 is entered that is executed until a termination criterion holds. In each iteration of the loop, a set of $\lambda = \lceil \mu \cdot \nu \rceil$ individuals is generated by applying genetic operators on the population. These new individuals are merged with the population to apply finally the selection operator which takes μ "best" individuals into the next generation. An interesting class of evolutionary algorithms are the evolutionary strategies that were introduced in the 1960s for the purpose of generating a set of rules for the automatic design and analysis of consecutive experiments (Beyer and Schwefel, 2002). Evolutionary strategies use specialized operators that were developed for real-valued optimization problems, even though there are variants that can be applied on integer optimization problems. This makes evolutionary strategies on the one hand

very powerful in their domain, on the other hand, however, these specialized operators do not allow for use on a wide domain of optimization problems. In

Algorithm 2.1: Loop of an Evolutionary Strategy

Input: function $f : X \to Y$

Output: $x \in X$ such that x is optimum of f

$t = 0$;

$P_t =$initialize(μ, σ, n_σ);

while (!terminated(P_t, t)) **do**

  $P'_t = \emptyset$;

  for $i = 1, \ldots, \lambda = \lceil \mu \cdot \nu \rceil$ **do**

    $P = \text{matingSelection}(P_t, \rho)$;

    $I = \text{recombination}(P, rec_x, rec_\sigma)$;

    $I' = \text{mutation}(I, c_\tau)$;

    $P'_t = P'_t \cup I'$;

  $P_{t+1} = \text{selection}(P'_t, P_t, \kappa)$;

  $t = t + 1$;

the following, all the genetic operators of the $(\mu, \nu, \rho, \kappa, n_\sigma, c_\tau, \sigma, rec_x, rec_\sigma)$-ES illustrated in Figure 2.1 are explained.

Individuals

Individuals are used to represent a potential solution of the problem under consideration. If a function $f : \mathbb{R}^n \to \mathcal{X}$ is to be optimized, hence the individual contains a vector $\boldsymbol{x} \in \mathbb{R}^n$ called the object component which gives a point of the search space. Moreover additional information is stored in the individual, namely the number of iterations the individual was part in the population in the form of an integer $\tilde{\kappa}$ (used for selection), the value $f(\boldsymbol{x})$ given the fitness of the individual (to omit multiple evaluations, hence to accelerate the optimization) and a strategy component that is either realized as a scalar $\sigma \in \mathbb{R}_+$ or a vector $\boldsymbol{\sigma} \in \mathbb{R}^n_+$ (used to steer the strength of the mutation). Hence an individual I is given as $I = (\boldsymbol{x}, \boldsymbol{\sigma}, f(\boldsymbol{x}), \tilde{\kappa})$.

Initialization

For initialization, different methods were proposed that are surveyed in (Beyer and Schwefel, 2002). In this thesis, especially the following method is appropriate: With a bounded search space, as this will be the case in the problems considered here, the population can be initialized uniformly at random within the search space. This is done by assigning to each coordinate i of each individual the value $[x]_i = a + \mathcal{U}_{[0,1]} \cdot (b - a)$, where $[a, b]$ is the domain of the i-th coordinate and $U_{[0,1]}$ is a uniformly distributed number in the interval $[0, 1]$. This initialization procedure has advantages especially on multi-modal optimization problems, where one expects several local optima. By spreading individuals over the whole search space, the probability of placing some individuals in sub-domains from which it is easier to reach the global optimum becomes much higher (Beyer and Schwefel, 2002).

The strategy component giving the step sizes for mutation is usually initialized by assigning the predefined value σ. Alternatively, the strategy component can be initialized by using a predefined interval $[\sigma_1, \sigma_2]$ from which values are drawn uniformly to initialize the strategy component. Two different types of strategy component are distinguished by the exogenous parameter n_σ, namely one step size for all dimensions, or alternatively one step size for each dimension, thus n step sizes.

Mating Selection

Mating selection $I^\mu \longrightarrow I^\rho$ is performed to choose ρ individuals from the population that are used to generate a new individual. The selection of individuals is based on randomness and requires that individuals having worse fitness do not have higher probability to be chosen than those individuals having better fitness. In the case of evolutionary strategies, mating selection is performed by choosing individuals uniformly at random.

Recombination

Evolutionary strategies distinguish between two recombination operators: Intermediate recombination and discrete recombination. The recombination operator maps the selected ρ individuals onto a new individual called the offspring. While the intermediate recombination calculates for a coordinate i the average over the parents coordinate i, the discrete recombination determines

the offsprings coordinate i by choosing coordinate i from a uniformly randomly selected parent. This procedure is repeated for all coordinates leading to a new individual. Formally, the recombination is hence realized as

$$[\boldsymbol{x}']_i = \frac{1}{\rho} \sum_{k=1}^{\rho} [\boldsymbol{x}^{(k)}]_i$$

in the intermediate case, and as

$$[\boldsymbol{x}']_i = [\boldsymbol{x}^{(\mathcal{U}_\rho)}]_i$$

in the discrete case, where $\boldsymbol{x}^{(j)}$ specifies the object component of the j-th individual and where the function $\mathcal{U}_\rho$ returns a uniform random number from the set $\{1, \ldots, \rho\} \subset \mathbb{N}$.

The recombination of the strategy component σ is performed analogously, moreover $\tilde{\kappa}$ is set to 0 in the newly generated individual, whereas a fitness evaluation is not performed in this step. The two exogenous parameters rec_x and rec_σ are used to specify the type of recombination for the object and strategy component.

Mutation and Self-Adaptation

The recombination operator does not ensure the complete exploration of the search space. This property however is a necessary condition to find the global optimum. Therefore, evolutionary strategies use a mutation operator that must be able to reach each point in the search space in finite time. Evolutionary strategies realize this by using the Gaussian distribution. Obviously, this distribution ensures that each point in the search space can be reached in finite time if numbers drawn from this distribution are added to each coordinate of an individual. Furthermore, they guarantee that mutation is symmetric, unbiased, and scalable (using different standard deviations), which are other conditions mutation has to fulfill. Concretely, the mutation of the object component is defined as

$$[\boldsymbol{x}']_i = [\boldsymbol{x}]_i + \mathcal{N}(0, [\sigma']_i) = [\boldsymbol{x}]_i + [\sigma']_i \cdot \mathcal{N}(0,1) \ ,$$

where $\mathcal{N}(0,\sigma)$ is a Gaussian distributed random number with mean 0 and standard deviation σ obtained from the strategy component.

The step sizes σ are obviously important for the success of a mutation. One can distinguish between two properties: The success-rate gives the number of

successful mutations and the progress-rate expresses the progress towards the optimum. Choosing $\sigma \to 0$ on the one hand will lead in 50% of mutations to a success, the progress towards the optimum however tends to zero due to very small movements in the search space. On the other hand, a large step size will not increase the progress since in most cases mutations will be unsuccessful. This phenomenon obviously depends on the state of the optimization. At the beginning, large step sizes are preferable in exploring the search space. At the end of the optimization, however, small step sizes are required to hit the optimum precisely. Hence a step size adaptation is required.

To realize step size adaptation, different methods were proposed, e.g., starting with a relatively high value and decreasing it with an increasing number of generations, adaptation according to the rate of successfully applied mutations, or self-adaptation techniques (Beyer and Schwefel, 2002). The latter are used in this work and will therefore be introduced more in detail. To realize self-adaptation the individuals store additional information in the form of the strategy component, which represents the standard deviation used for mutation. Keeping in mind that standard deviations are numbers larger than zero and the properties that mutation has to fulfill (e.g. symmetry), multiplication with a logarithmic normally distributed number is the most appropriate approach to mutate the strategy component. To allow different adaptation rates, an exogenous parameter c_τ is used to define $\tau = c_\tau / \sqrt{2\sqrt{n}}$ and $\tau_0 = c_\tau \cdot \exp(\mathcal{N}(0,1)) / \sqrt{n}$. These two values are used to realize the mutation of the strategy component σ by

$$[\sigma']_i = [\sigma]_i \cdot \tau_0 \cdot \tau \cdot \exp(\mathcal{N}(0,1)) \, .$$

Since there is no fitness function for the strategy component, it is evaluated indirectly. The idea is firstly to mutate the strategy component and afterwards the object component using the standard deviations stored in the already mutated strategy component, followed by the subsequent evaluation of the thus resulting individual. The assumption is that the better the strategy component is, the better the object component becomes. Thus, better strategy components will lead to a higher fitness that in turn leads to a higher probability for the selection of the corresponding individual. Hence, individuals having better strategy components are more likely to be reproduced.

Selection

The selection operator reduces the surplus of individuals generated by the operators recombination and/or mutation, to ensure that the population P_t contains μ individuals. In this work, the κ selection is considered. To realize this selection operator, each individual contains an integer $\tilde{\kappa}$ called age, giving the number of generations it already exists in the population. Having a set of m individuals, the selection operator choses the best μ individuals for the next generation that do not exceed an age of κ. This generalized operator allows one to model the comma-selection and the plus-selection as well, since for choices $\kappa = 1$ and $\kappa = \infty$ the former and the latter are realized, respectively. Furthermore it allows for finding a trade-off between both extremes, where in the former case it was observed that an evolutionary strategy does not converge to an optimum and in the latter case that it is more likely to get stuck in a local optimum.

Termination Criteria

In contrast to classical algorithms, it is hard to determine for evolutionary algorithms whether the global optimum (at least approximately) has been found. Different techniques exist: Some of them do not use criteria based on the quality of the solution found so far and terminate the evolutionary loop if a certain number of generations or fitness evaluations was performed, or if a certain amount of time was used. If it is sufficient to reach a certain quality, the best fitness value so far can be monitored and the loop is terminated after reaching the specified quality. Criteria that are based on the progress of the optimization often use the convergence velocity or the number of stall generations or time. Here, the assumption is that the optimum is probably reached if after a certain amount of time or generations the best fitness could not be increased. Accordingly, if the convergence velocity reaches a certain predefined value, again the hit of the optimum can be assumed. Using the self-adaptation technique, another interesting termination criterion can be used based on the current step size. The step sizes decrease with increasing progress of the optimization. Therefore, the search can be terminated if the step size falls below a certain threshold.

2.4 Similarity, Distance and Score

Three types of functions are considered in this thesis: *Similarities*, *distances* and *scores*. Generally, similarities and distances are binary functions, mapping two objects onto a real number. This number indicates in the former case the similarity degree between both objects, in the latter case the distance degree. Obviously, both functions are related, hence a distance can be transformed in different ways into a similarity and vice versa (Deza and Deza, 2009). Distances and similarities can exhibit different properties. A special class of distance is the so-called metric $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ which fulfills identity of indiscernibles, symmetry and the triangle inequality, that are defined as follows:

identity of indiscernibles: $d(x, x') = 0$, if and only if $x = x'$,

symmetry: $d(x, x') = d(x', x)$,

triangle inequality: $d(x, x') \leq d(x, x'') + d(x'', x')$.

Given these three properties, it moreover follows that a metric is obviously non-negative. Another class are the so-called kernel functions $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, requiring symmetry and positive-definiteness, that is

$$\sum_{i,j=1}^{m} c_i c_j k(x_i, x_j) \geq 0$$

for all $m \in \mathbb{N}$, $\{c_1, \ldots, c_m\} \subseteq \mathbb{R}$ and $\{x_1, \ldots, x_m\} \subseteq \mathcal{X}$. The (usually unnormalized) value a kernel returns corresponds to the similarity of the two objects considered. Hence, a kernel can be considered as a similarity measure allowing for a direct usage in retrieval and classification as done in this thesis. However, there exist different mappings to normalize a kernel k or to transform it into a metric d. In (Riesen and Bunke, 2010) the authors show that any dot product $\langle \cdot, \cdot \rangle$ in the Hilbert space (Bronstein et al., 2008) is a kernel. Moreover, there exist a relation between the dot product in the Hilbert space and the norm, namely $\|x\| = \sqrt{\langle x, x \rangle}$. Given this information, a way to derive a normalized kernel $\tilde{k}$ is to calculate

$$\tilde{k}(x, x') = \frac{k(x, x')}{\sqrt{k(x, x) k(x', x')}} ;$$

to derive a distance d which moreover fulfills the metric properties, the following mapping can be used:

$$d(x, x') = \sqrt{k(x, x) + k(x', x') - 2k(x, x')} .$$

The term score refers to another concept, which is in combination with evolutionary algorithms often called fitness. A score is a function mapping an element of a search space usually onto a real number. For some similarity measures proposed in this thesis, the solution of optimization problems becomes necessary. Then, similarity corresponds to the maximal score which can be reached. Moreover, a score can be defined on a set of more than two objects which becomes interesting when a whole family of proteins is analyzed, which usually consists of much more than two proteins.

2.4.1 Generalizing Similarity

Similarity can be defined in different ways: One way to realize similarity is based on a strict equivalence in which the structures must match perfectly to obtain the maximal similarity value. Such measures, however, are not fully appropriate, especially if the two structures greatly differ in size. In some applications, it makes sense to have a high similarity degree even if structure x is only a substructure of x', for example if x is a subpocket of x' containing the most important catalytic residues (while the rest of the binding site x' is functionally less important). Keeping in mind, that equivalence of sets is defined as

$$x = x' \Leftrightarrow x \subseteq x' \ \wedge \ x' \subseteq x \, ,$$

similarity can be defined in a relaxed way by considering similarity as a fuzzy equivalence which can be realized by two fuzzy inclusions. An interesting generalization, therefore, is to let

$$\begin{aligned} \mathrm{sim}(x, x') = & \lambda \cdot \min\{\mathrm{inc}(x, x'), \mathrm{inc}(x', x)\} \\ & + (1 - \lambda) \cdot \max\{\mathrm{inc}(x, x'), \mathrm{inc}(x', x)\} \end{aligned} \, . \tag{2.1}$$

In the case that the function inc returns values in the unit-interval, this similarity measure can be motivated from a fuzzy logical point of view as follows: Considering the min (max) operator as a generalized conjunction (disjunction), the first (second) combination of the two inclusion degrees is the truth degree of the proposition that x is contained in x' AND (OR) x' is contained in x. A conjunctive combination of the two degrees of inclusion is obviously more demanding than a disjunctive one, as the former requires equality between x and x' while the latter only requires inclusion of x in x' or x' in x. The measure (2.1), which formally corresponds to an *ordered weighted average* (OWA)

combination of the two degrees of inclusion (Yager, 1988), achieves a trade-off between these two extreme aggregation modes, which is controlled by the parameter $\lambda \in [0,1]$: The closer λ is to 0, the closer the aggregation is to the inclusion, i.e., the less demanding it becomes. The optimal λ is application-specific and depends on the purpose of the similarity measure.

2.5 Structural Alignments and Conserved Patterns

So far functions were discussed which can be used to retrieve similar structures from CavBase. Sometimes, however, one is interested in more than the similarity value, e.g. in the reasons why a certain similarity value was obtained. Here alignments and conserved patterns can help. Moreover, both are appropriate to define a similarity between protein binding sites, e.g., in terms of the fraction of the size of the conserved pattern and the size of the structures.

2.5.1 Structural Alignment

Lets assume to have a set of structured objects $\{X_1, \ldots, X_m\} = \mathcal{X}$, where each $X \in \mathcal{X}$ is represented by a finite set of elements $\{x\}$ (e.g. a set of point clouds $\mathcal{P}$, where each point cloud $P \in \mathcal{P}$ consists of a set of n points $x \in \mathbb{R}^3$). Roughly speaking, the result of the alignment of the structures in $\mathcal{X}$ is an one-to-one correspondence between the elements of all $X \in \mathcal{X}$. Finding a one-to-one correspondence obviously requires equal-sized sets X which are not given in practice. Hence, it is absolutely necessary to include dummy elements $\perp$ into a set X to be able to find a one-to-one correspondence. Dependent of the size of $\mathcal{X}$, an alignment is called pairwise ($m = 2$) or multiple ($m > 2$). Formally, the structural alignment is defined as

Definition 2.17 (Structural Alignment)

Let $\mathcal{X} = \{X_1, \ldots, X_m\}$ be a set of structured objects represented as described above. Then $\mathcal{A} \subseteq (X_1 \cup \{\perp\}) \times \cdots \times (X_m \cup \{\perp\})$ is an alignment of the objects in $\mathcal{X}$ if and only if the following two properties hold:

1. *Each substituent of each object occurs exactly once in the alignment, i.e., for all $i = 1, \ldots, m$ and for each $x \in X_i$ there exists exactly one $a = (a_1, \ldots, a_m) \in \mathcal{A}$ such that $x = a_i$.*

2. *Each tuple of the alignment contains at least one non-dummy element, i.e., for each* $a = (a_1, \ldots, a_m) \in \mathcal{A}$ *there exists at least one* $1 \leq i \leq m$ *such that* $a_i \neq \perp$.

The Definition 2.17 obviously allows for an enormous number of valid alignments and one is looking for the optimal. The most trivial way to find the optimal alignment is to use a scoring function and to develop an optimizer returning that alignment with best score. However, in some cases it makes sense to use other techniques, since the size of the search space of the multiple structural alignment problem is $\mathcal{O}(((|X_1| + \cdots + |X_m|)!)^{m-1})$, hence it grows super-exponential with the number and size of structures. This size is of course problematic from an optimization point of view. Obviously the efficiency of a search algorithm depends on the size of the search space, hence efficiency can be increased by down-scaling the search space. One established strategy to reduce the search space is to use decomposition techniques. Here the optimal multiple alignment problem is decomposed into a set of pairwise alignment problems. Subsequently, after solving these presumably more simple pairwise problems, composition techniques are applied to merge the pairwise alignments to a multiple alignment. Two interesting and often used techniques to realize this concept are *star-* and *tree-alignment*. The difference to calculating the multiple alignment at once obviously is that flexibility is lost. Alignment can always be regarded as an edit-sequence. Considering the alignments obtained as such a transformation rule, in the original problem formulation the problem is roughly speaking defined as *"find a cost-minimal edit-sequence to transform the structures to an arbitrary new structure"*. Using decomposition techniques it becomes *"find a cost-minimal edit-sequence to transform the structures to one of the input-structures"*.

Thus, decomposition techniques are a purely heuristic aggregation procedure, improved efficiency is likely to come with a decrease in solution quality, compared with an approach which would solve the multiple alignment problem at once. This is not necessarily the case, however. In fact, a decomposition essentially produces two opposite effects, a positive one due to a simplification of the problem and, thereby, a reduction of the search space, and a negative one due to a potentially suboptimal aggregation of the partial solutions. For a concrete problem, it is not clear in advance which among these two effects will prevail. Roughly speaking, it is very possible that constructing good pair-

wise alignments and aggregating them in an ad-hoc way is better than getting astray in a huge search space of multiple alignments, especially because the multiple structural alignment problem is known to be NP-complete and one cannot expect an efficient and exact solution to this problem.

Star-Alignment

Star-alignment is a decomposition technique that allows one to calculate a multiple alignment of m structures by solving $(m-1)$ pairwise alignment problems and merging them. One of the structures, say, X_1, is selected and aligned in a pairwise fashion with all other structures X_i, $i = 2, \ldots, m$. The pairwise alignments are then merged by using X_1 as a pivot structure. Thus, if $x_i^j \in X_i$ denotes the substituent (possibly a dummy) aligned with $x_1^j \in X_1$ in the alignment of X_1 and X_i, then a single assignment in the multiple alignment is of the form

$$a = (x_j^1, x_j^2, x_j^3, \ldots, x_j^m) \ . \tag{2.2}$$

If more than one dummy was inserted into the pivot structure, resulting assignments are not unique. To increase quality one could formulate another optimization problem which, however, would make the overall approach more expensive. Therefore, a heuristic is used where the pairwise assignments containing a dummy in the pivot structure are merged randomly.

Since the quality of a multiple alignment thus defined is strongly influenced by the choice of the pivot structure, each structure is tried as a pivot and the best result is adopted. Thus, $m \cdot (m-1)/2$ pairwise alignments have to be computed in total. Figure 2.3 shows an example of this procedure. Here structure X_1 was chosen as the pivot element. In the first step the pairwise alignments of the pivot structure X_1 and the remaining structures X_2, X_3 and X_4 are constructed (that are represented in the form of a matrix in which each column corresponds to an assignment). Subsequently, the pairwise alignments are sorted according to the indices of the substituents of the structure X_1. After sorting, multiple alignment can be constructed by simply copying the cells of the pairwise alignments into the corresponding position in the multiple alignment. Those cells that occur in the multiple alignment but in any pairwise alignment are filled with dummys.

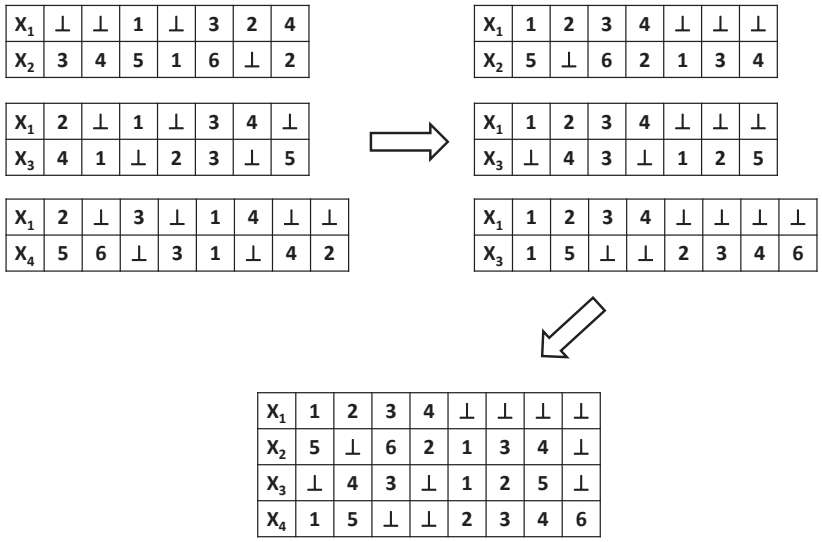


Figure 2.3: Visualization of the three steps of the star-alignment procedure in which the first structure is used as a pivot: First pairwise alignments are constructed, followed by a sortation of the columns according to the pivot and the subsequent merging of the pairwise alignments to a multiple alignment.

Complexity Constructing multiple alignments of size m by using the star-alignment procedure requires the calculation of $\mathcal{O}(m^2)$ pairwise alignments. Hence, the overall complexity of this step depends mainly on the complexity C of the method used to calculate these pairwise alignments. The subsequently applied merging procedure mainly consists of the sorting of indices, hence a complexity of $\mathcal{O}(|\mathcal{A}| \log(|\mathcal{A}|))$ is obtained for each considered pairwise alignment $\mathcal{A}$. In sum, since C is usually quite high, the whole approach therefore has a complexity of $\mathcal{O}(m^2 \cdot C)$. The space complexity is given by $\mathcal{O}(n \cdot m)$, where $n = |X_1| + \cdots + |X_m|$, since enough space is required to store the matrices used to represent the multiple and pairwise alignments. This low complexity, however, will usually be dominated by the space complexity of the algorithm constructing pairwise alignments.

Tree-Alignment

With tree-alignment, another procedure is available in which the selection of a pivot structure becomes obsolete. To this end, a tree-based approach for sequences taken from (Wheeler and Kececioglu, 2007) is adapted for structures:

As a first step, an UPGMA[1]-tree (Sokal and Michener, 1958) is constructed in which the m structures are used as leaves. To calculate such a tree, the pairwise distances between leaves are required. Here, a measure on structures (distance or similarity) is sufficient and allows for the construction of such a tree. In each step of the bottom-up construction of the tree, a node is inserted to the tree that becomes the parent of those two parent-less nodes having the highest similarity among all parent-less nodes. After the insertion of the node, the pairwise similarities must be calculated to all other nodes that have no parent so far. Since an inserted node $\mathcal{A}$ represents an alignment of those leaves that are reachable from $\mathcal{A}$, it makes sense to define similarity to another parent-less node $\mathcal{A}'$ by

$$\frac{1}{|\mathcal{A}| \cdot |\mathcal{A}'|} \cdot \sum_{X \in \mathcal{A}} \sum_{X' \in \mathcal{A}'} s(X, X') \,, \tag{2.3}$$

hence, similarity between $\mathcal{A}$ and $\mathcal{A}'$ is obtained by averaging the similarities between the structures aligned in $\mathcal{A}$ and $\mathcal{A}'$. Figure 2.4 shows an example of

	1	2	3	4	5	6	7	8	9
1	1	.4	.8	.2	.6				
2		1	.6	.7	.5	.5			
3			1	.2	.2				
4				1	.5	.2			
5					1	.4	.5		
6						1	.4	.4	
7							1		
8								1	
9									1

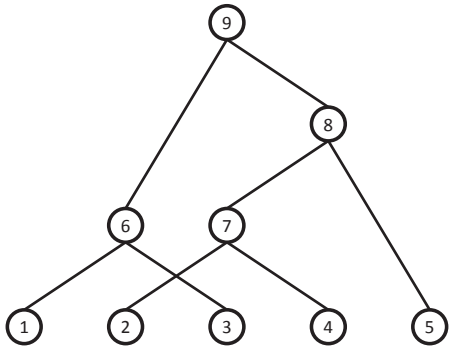


Figure 2.4: Example for an UPGMA-tree: Five structures are clustered according to their pairwise similarities. Note that similarities are given without integer-place 0 in the table.

the procedure in which five structures are considered. The pairwise similarities are given in the sub-matrix emphasized by the dotted-line. The most similar structures are indexed by 1 and 3, hence, they are merged in a node indexed by 6. The similarities of the new node to all parent-less nodes (indexed by 2,4,5) are calculated according to (2.3). The most similar structures represented by parent-less nodes are, however, 2 and 4 that are merged in a new node 7. Again, similarities between the new node and remaining parent-less nodes 6

[1]UPGMA is the abbreviation for unweighted pair group method with arithmetic mean.

and 5 are calculated. The procedure is repeated until all nodes are merged and a root is formed representing the complete multiple alignment.

The idea behind this procedure is that it makes more sense to align similar structures and to merge similar alignments (to a larger multiple alignment) since such alignments share as a rule more structural similarity which can be exploited during merging, leading to better alignments. Finally, more dissimilar alignments are merged. However, due to the already reached size of the alignments to merge, they behave more stably, hence it is not likely that good alignments should drop very much in quality.

Hence, after construction of the UPGMA-tree, the multiple alignment is created according to the tree, which defines the bottom-up order in which the alignments are merged. The pairwise alignments obtained by merging leaves are calculated by applying a pairwise alignment solver. Merging two inner nodes, each of which corresponds to multiple assignments $a^i = (a^i_1, \ldots, a^i_n)$ and $b^j = (b^j_1, \ldots, b^j_m)$, respectively, is accomplished by calculating the averaging pairwise distances between assignments:

$$d(a^i, b^j) = \sum_{k=1,..,n} \sum_{l=1,..,m} d(a^i_k, b^j_l)\,, \tag{2.4}$$

where $d(a^i_k, b^j_l)$ is the distance between substituents a^i_k and b^j_l. Having calculated all these distances, again a pairwise alignment problem is obtained which can be solved with the pairwise solver. This procedure can be easily adapted to the case where an inner node is merged with a leaf by simply removing one (of the two) sums. At the end of the procedure, in the root of the UPGMA-tree the multiple alignment of the m structures is finally constructed.

Complexity The construction of multiple alignments by using the UPGMA-tree takes polynomial runtime. For construction of the UPGMA-tree, a complexity of $\mathcal{O}(m^2)$ is obtained, where m is the number of leaves, thus structures. The UPGMA-tree, which is binary, has at most $\mathcal{O}(m)$ inner nodes. Hence, $\mathcal{O}(m)$ (pairwise) alignments must be constructed. Moreover, the pairwise distances between assignments must be updated according to eq. (2.4) in each inner-node. Since the length of an alignment is $\mathcal{O}(nm)$, where n is defined as $\max_{i=1,..,m} |X_i|$, and because all assignments are considered pairwise, the runtime for these steps becomes in sum $\mathcal{O}(nm^3)$. However, the complexity is clearly dominated by the approach needed to construct the pairwise align-

ments which is in turn based on the method calculating optimal superpositions. Since this method is coming with a complexity C which is usually much higher than $\mathcal{O}(nm^3)$, the overall complexity of this method will be $\mathcal{O}(n \cdot C)$.

The space complexity is also growing in comparison to the former approach. Since alignments of size $\mathcal{O}(nm^2)$ must be stored in each inner-node, and because there are $\mathcal{O}(m)$ such nodes, the overall space complexity is $\mathcal{O}(nm^3)$.

m-partite Graph Matching

Knowing the costs caused by assigning substituents, the optimal way to calculate multiple geometric alignments would be to solve the m-partite graph matching problem. This problem is specified analogously to that of bi-partite graph matching (cf. (Corman et al., 2001) or Section 4.2), namely by a graph $G = (V, E, \ell_E)$, where its set of nodes is decomposed into m disjunct sets

$$V = \bigcup_{i=1}^{m} V_i \ ,$$

and where the set of edges is given by

$$E = \left\{ (v_i, v_j) \mid v_i \in V_p,\ v_j \in V_q,\ p, q = 1, \ldots, m,\ p \neq q \right\} \ ,$$

with costs $\ell_E(e)$ associated with each edge $e \in E$. The goal is to find a subset of edges $M \subset E$ that minimizes $\sum_{e \in M} \ell_E(e)$ and fulfills the requirement that each node of each partition is adjacent to exactly one node of each other partition.

In the m-partite graph a partition V_i corresponds to the substituents $x \in X_i$, accordingly the costs assigned to edge $(v_i^{(k)}, v_j^{(l)}) \in E$ to the cost caused when substituent $x_k \in X_i$ is assigned to substituent $x_l \in X_j$. To account for substituent-to-dummy mappings a partition V_j moreover contains $|X_1| + \cdots + |X_{j-1}| + |X_{j+1}| + |X_m|$ dummys. The costs for assigning a substituent to a dummy are specified by a parameter k, the costs for assigning a dummy to another dummy by zero, so that the latter mappings will not influence the construction of the alignment. Solving the m-partite graph matching problem on this kind of input would allow one to calculate the multiple alignment without taking the indirect way of first solving a number of pairwise alignments followed by merging them, e.g., in terms of a star-alignment procedure. This technique however would become inefficient for larger inputs since the problem of m-partite graph matching is known to be NP-complete (Hazan et al.,

2003). Shatsky et al. (2006) therefore proposed a modification, namely an m-partite *pivot* graph for solving the multiple graph alignment problem. An m-partite pivot graph is a slight modification of an m-partite graph in which another set E is considered. While in the original approach all pairs of nodes, each from a different partition, are adjacent to each other, in the m-partite pivot graph there exist all edges that are adjacent to a node in the m-th partition and another node not contained in the m-th partition. Thus the set of edges E becomes much smaller and is defined as

$$E = \left\{ (v_i, v_j) \mid v_i \in V_{pivot}, v_j \in V_q,\ q = 1, \ldots, m,\ q \neq pivot \right\}.$$

Accordingly, the optimization problem becomes simpler and can be solved in polynomial time, namely by applying a greedy heuristic, a process that consists of n iterations, where $n = |V_1| = \cdots = |V_m|$, depicted in Figure 2.2. In the i-th iteration of the algorithm, $v_i \in V_{pivot}$ is considered. For each partition V_j dif-

Algorithm 2.2: m-Partite Pivot Graph Matching Solver

```
Input: m-partite pivot graph
Output: m-partite matching as a set of m-tuples of nodes
initialize list matching;
for v_i ∈ V_pivot do
    initialize ordered list l;
    for j = 1, ..., m do
        if j = pivot then
            l.add(v_i);
            continue;
        v_s = arg min_{v_k ∈ V_j}({ℓ_E(v_i, v_k)});
        for v_k ∈ V_pivot do
            ℓ_E((v_s, v_k)) = ∞;
        l.add(v_s);
    matching.add(l);
return matching;
```

ferent from V_{pivot}, such edges are considered that connect the nodes in V_j with the node v_i. From this set $\tilde{E} = \{(v_i, v_k) \mid v_k \in V_j\}$, the edge (v_i, v_s) is selected for which $\ell_E((v_i, v_s)) < \ell_E((v_i, v_k))$ for all $v_k \in V_j$, hence the node v_s becomes

part of the assignment. Since each node can participate in exactly one assignment, all edges in $\tilde{E}$ are excluded from further consideration. This procedure is continued until all nodes in V_{pivot}, therefore all nodes in the other partitions, are processed. The set of the constructed m-tuples, each of which represents an assignment, is returned as the optimal multiple structural alignment.

In comparison to star-alignment, this technique has on the one hand the advantage that the complete multiple alignment can be calculated at once without decomposition into pairwise alignments. On the other hand, solutions obtained for the m-partite pivot graph matching problem are not necessarily optimal compared to such a solution obtained by applying a method solving the general case in which m-partite graphs are considered. As in the case of star-alignment, the solution obtained obviously depends of the choice of the pivot structure. Therefore, the procedure is repeated m times, where in the i-th iteration the i-th structure is used as pivot, and the best solution is returned.

Complexity The size of each partition V_i ($i = 1, \ldots, m$) is given by $n = |X_1| + \cdots + |X_m|$. The algorithm performs n loops, where in the i-th cycle all partitions different from the pivot and the i-th node of the pivot partition $v_p^{(i)}$ are considered. For a partition V_j different from the pivot the list $(l_E(v_p^{(i)}, v) \mid v \in V_j, v \text{ not yet assigned})$ is retrieved and the minimum is identified in time $\mathcal{O}(n)$. Doing this for all partitions different from the pivot and for all nodes in the pivot leads to a runtime of $\mathcal{O}(n^2 m)$. Since each partition is tried as pivot the overall complexity is given by $\mathcal{O}(n^2 m^2)$. The space complexity is given by the size of the matrix storing the m-partite pivot graph which is $(m-1)n^2$. However, again one should note that an additional approach must be applied to determine the costs for assignments of substituents. The algorithm required for this step will usually have a much higher complexity.

2.5.2 Conserved Pattern

Having established a multiple alignment it is of interest to identify the so-called conserved patterns, since these patterns provide information about the evolutionary heredity or may represent substructures existing in all proteins of a family of proteins responsible for their function. The conserved pattern can be defined in a similar way as that for the alignment of sequences, since the structural alignment is still given in the form of a matrix (Weskamp et al., 2007).

The main difference is that substituents of a structure are not ordered as characters of a string. Hence, non-connected columns can also contribute in one conserved pattern which is defined by two functions:

$$cons(a) = \frac{|\{i \mid a_i \neq \perp\}|}{m} \tag{2.5}$$

$$maj(a) = \max_{l \in \mathcal{L}} \frac{|\{i \mid a_i = l\}|}{|\{i \mid a_i \neq \perp\}|} \,. \tag{2.6}$$

On the one hand, the function (2.5) returns the degree of conservation for an assignment a, which is the relative number of mutually mapped substituents different from dummy. The function (2.6) on the other hand is returning the relative number of substituents different from dummys labeled with the most frequent label in a. By defining two thresholds $\omega, \xi \in (0,1]$, giving respectively the minimal degree of $cons(a)$ and $maj(a)$ required to call an assignment a conserved, the conserved pattern can be retrieved, namely by selecting those assignments $a \in \mathcal{A}$ which pass the two constraints.

3

Related Work

This chapter gives an overview on related work. Parts of this chapter were already published in (Fober et al., 2009b) and (Fober et al., 2011).

Proteins are often represented in the form of strings, i.e. sequences of amino acids. To measure similarity between strings and to analyze them e.g. in terms of alignments therefore quite a number of methods have been proposed. The focus of this thesis, however, lies on the structural comparison and analysis of molecules, due to the reasons mentioned in Chapter 1. Therefore, methods developed for the sequence-based analysis of proteins will be not considered here, instead one is referred to (Chao and Zhang, 2009) which give an excellent overview on these methods.

The structural comparison of proteins has long been a central task in bioinformatics. Moreover, the problem of identifying common substructures among structured data has simultaneously arisen from many other fields of research, including chemoinformatics, pattern recognition, data mining, database systems and many more. Often, the resulting algorithms can also serve as methods for the comparison and analysis of proteins. Hence, theoretically an enormous number of methods exist to compare and to analyze proteins.

In particular, the use of graphs as a modeling concept for structured data has been proposed by several authors. Since graphs are a rather general data structure, graph-based methods are very flexible and widely applicable. In the field of bioinformatics, for example, graphs have not only been used for modeling molecular structures, but also for modeling biological networks, such as regulatory networks (Davidson et al., 2002), interaction networks (Berg and Lässig, 2004; Xenarios et al., 2002), metabolic networks (Kanehisa et al., 2004), or phylogenetic networks (Huson and Bryant, 2006). Moreover, graph-based

models also play an important role beyond the domain of bioinformatics. For example, graphs can be used to model other kinds of networks, such as social networks (Wassermann and Faust, 1994), HTML / XML documents (Page et al., 1998), or the internet itself (Borgwardt, 2007).

Another class of modeling concept is sets of points in the Euclidean space, where points are enriched with additional information. Surprisingly, this representation is not often used for the analysis of proteins. Instead, it is applied in image processing, pattern recognition, cartography, industrial inspection and robotics (Thompson et al., 1999; Fabio, 2003; Höfle et al., 2007; Irfanoglu et al., 2004; Munoz et al., 2008), though it has some advantages compared with graphs: Data is often given as a set of observations that are distributed in the Euclidean space, hence a set of points becomes the natural representation guaranteeing no loss of information that can occur during transformation from one representation to another one. Moreover, processing on sets of points may allow more efficient calculations since the combinatorial character of many algorithmic problems on graphs that often lead to NP-hard problems usually does not appear for algorithms operating on point sets.

Obviously, both representations, graphs and point clouds as well, are appropriate to model protein binding sites, hence algorithms able to process on these representations known from literature are introduced in the following. Besides general algorithms which operate on arbitrary graphs and arbitrary point clouds, specialized methods are presented that exploit the additional information given by proteins.

3.1 Geometric Approaches

Point cloud data is often generated by laser scanners or represents objects captured by cameras, e.g., a product that is matched against the prototype to determine its quality. Such techniques as a rule generate a large set of points, hence methods which allow for processing such data are often very efficient, mostly taken from the field of computational geometry. However, in most cases these methods are designed for a certain application, a reason why they are often limited to certain transformations (e.g. translation in $\mathbb{R}^2$ instead of rotation and translation in $\mathbb{R}^3$). Moreover, since in this area one is mostly interested in the geometry of an object or image, labels are not considered. Hence, almost all methods are developed for the case of unlabeled point clouds, and therefore

must be extended before being applied on protein binding sites.

For point clouds, different measures were proposed, some that are exact measures often combined with the ability to establish a one-to-one mapping, thus an alignment of points. Other methods, which originate from the need of allowing a certain error-tolerance are often based on the Hausdorff-distance, a standard concept to measure the distance between two subsets of a metric space. Other works use geometric hashing or ϵ-neighborhoods, both especially for the calculation of an approximate alignment. Measures establishing correspondences between points are usually quite expensive. An interesting approach is therefore to transform a set of points into a feature vector and to compare pairs of feature vectors afterwards. A promising way for doing this was presented by Kupas et al. (2007) who used wavelet functions for this purpose. The point set was decomposed into circular patches and wavelet functions approximating the patches were fitted. The resulting coefficients were subsequently used to describe the geometry. This approach can be easily adapted to labeled point clouds by extending the feature vector, e.g., by the percentage of points within a patch exhibiting a certain label, as proposed by the aforementioned authors.

3.1.1 Exact Point Matching

Interesting methods following the exact matching concept were developed by Atkinson (1987); Alt et al. (1988); Sprinzak and Werman (1994). In (Atkinson, 1987) an efficient technique is used that reduces the problem of point matching to the problem of string matching for which efficient algorithms are known (Chao and Zhang, 2009). This technique, however, considers point clouds in two dimensions only. The extension by Alt et al. (1988) considers planar graphs instead of strings and calculates an isomorphism of these planar graphs by using an efficient algorithm (Hopcroft and Wong, 1974), leading to a matching of 3-dimensional point clouds. Also in (Atkinson, 1987), coordinate vectors are used to find an exact matching of points in three dimensions. Sprinzak and Werman (1994) use canonical forms for this purpose. Generally, exact point matching is comparable to graph isomorphism, hence suffers from the same problems: Although exact point matching can be calculated efficiently, it is not appropriate as a measure for protein binding sites since these concepts allow absolutely no error-tolerance and do not take labels into consideration. A

promising alternative to exact point matching therefore is approximate point matching.

3.1.2 Approximate Point Matching

The problem of approximate point matching is typically defined as the minimal number ϵ so that after proper translation and rotation, each point in the first cloud has a counterpart in the second cloud that fall into its ϵ-neighborhood, and vice versa. Obviously, the minimal ϵ can serve as a distance measure between point clouds. Different concepts were introduced, those that guarantee that each point has exactly one counterpart and those that allow a point to match more than one point in the other set. In the latter case the Hausdorff distance is often used as a distance measure.

To solve the one-to-one correspondence problem, Alt et al. (1988) combined algebraic curves traced by certain points with an algorithm solving the bipartite graph matching problem (Kuhn, 2005). This approach unfortunately suffers from its complexity which is polynomial of order eight. More efficient approaches are given by Efrat and Itai (1996) and Arkin et al. (1992) that require, respectively, that only translation is considered or that the ϵ-neighborhood regions are disjoint. Decision and approximate decision algorithms for this kind of problem are given in (Heffernan and Schirra, 1992) which solve the problem e.g. with network flow algorithms (Corman et al., 2001). The concepts presented here seem very interesting since they allow the calculation of a distance between point clouds and moreover an alignment that is given by the calculated correspondences. Unfortunately, all these concepts come with high complexity that can be reduced only by making assumptions that will not allow use of the resulting methods on protein binding sites. Other problems of these methods are the requirement of equal-sized point clouds and the lack of an ability to consider label information.

Algorithms that are based on the Hausdorff distance do not require point sets of equal cardinality, hence are in this regard more appropriate for the purpose of a protein binding site comparison. Moreover, the calculation of this distance can be performed very efficiently using Voronoi-diagrams (Alt et al., 1991). Unfortunately, the Hausdorff distance is not invariant against translation and rotation, thus an optimal transformation must be calculated for sets having different origin. Huttenlocher et al. (1993a) use properties of Voronoi-

diagrams to calculate an optimal transformation, however, to calculate an optimal rotation, dynamic Voronoi-diagrams must be used that lead to a polynomial complexity of order six. Modifications of this approach consider the one-way Hausdorff distance, hence seek partial matches (Huttenlocher et al., 1993b). As Alt and Guibas (1996) notice, these approaches are numerically unstable and hard to implement, which makes a further extension to labeled points very difficult. More promising algorithms were proposed by Goodrich et al. (1994); Hoffmann et al. (2010): While the former is an approximation of the Hausdorff distance with approximation factor eight, the latter defines its own distance measure (different from the Hausdorff distance) and optimizes it with gradient descent methods to find the optimal translation and rotation. This approach already takes point labels into account, hence, it can be used directly for the comparison of protein binding sites.

Another interesting approach, namely geometric hashing, allows calculation of a partial alignment between point clouds independent of a certain distance measure (Shatsky et al., 2006; Bachar et al., 1993; Leibowitz et al., 1999; Lamdan and Wolfson, 1988; Wolfson and Rigoutsos, 1997; Leibowitz et al., 2001). These approaches have in common the use of a hash-table, that contains k-tuples drawn from a certain point set. This hash-table is used in the recognition phase, where k-tuples of the other point sets are drawn and looked up in the table. If two matching k-tuples are found, one from the first and one from the second point set, they can be superimposed and thus define a *transformation* that can be used to derive an alignment. This approach allows one to enrich the points with additional information. Moreover, it allows one to calculate multiple alignments. Unfortunately, the hash-table can become quite large, which often leads to a high runtime and additionally to a failure of the whole approach on large inputs due to a memory overflow.

In (Wang and Wang, 2000), point clouds are transformed into 3D graphs and hashing is applied for a fast similarity search. In a recent paper, Bach (2008) proposes transforming a point cloud into a graph and applying kernels on such graphs afterwards, an approach that has already long been used, e.g. in bio- and chemoinformatics (Borgwardt et al., 2005).

3.1.3 Superposition based on One-to-One Correspondences

If the one-to-one correspondence between points is known and the goal is to find the optimal superposition of point sets, different algorithms can be used. A well-known method often applied in chemoinformatics is described in (Kabsch, 1976) which minimizes the root mean squared deviation of two point sets by using simple matrix operations. Another approach analyzes the lower envelope of multivariate functions, hence functions of more than one variable, to superimpose two point sets (Imai et al., 1989). At first sight, such methods seem worthless since they already require alignment. However, such tools can be used e.g. to improve the score obtained from a (partial) alignment as done in CavBase. Moreover, some approaches require such techniques as a sub-procedure. An example is the geometric hashing approach that looks for approximately equivalent *k*-tuples that are used to determine the transformation.

3.2 Graph-based Approaches

A couple of generic principles of graph similarity can be distinguished on which most of the existing approaches are based. A first concept that has been widely used in chemoinformatics, pattern matching and computer vision considers two graphs similar if they are isomorphic or share at least a common subgraph which leads to the (sub)graph isomorphism problem, and closely related to this, the concept of the maximum common subgraph. Instead of looking for a single, as large as possible compliance, one may also look for many small compliances. Thus, it may be more reasonable to look for a large number of smaller common substructures and to define similarity accordingly, which is the basic idea of frequent subgraph mining. A third principle is based on the generic concept of an edit distance. According to this principle, two graphs are similar if a few modifications, so-called edit operations, are sufficient to make the first isomorphic to the second. In contrast, the first two approaches focus primarily on exact matches between graphs. Although it is possible to extend these concepts to approximate similarity, such extensions are often difficult to realize algorithmically. Other approaches aim at representing graphs by defining certain representative features and calculating similarity accordingly. Since the generation of features is not restricted to graph representations, such

techniques can also be adapted to many other representations as labeled point clouds.

3.2.1 Methods based on Graph Isomorphism

Graph isomorphism and subgraph isomorphism are standard concepts for determining the similarity of graphs in the field of pattern matching, for which standard algorithms have long been known (Ullmann, 1976; Read and Corneil, 1977; Hopcroft and Wong, 1974). Closely related to these concepts is the principle of common subgraphs. In chemoinformatics, the concepts of maximum common subgraph (Bunke and Jiang, 2000) and minimum common supergraph (Bunke et al., 2000) have been widely used for the comparison of chemical compounds (Raymond and Willett, 2002). Using e.g. the maximum common subgraph (MCS), a similarity measure can be easily defined, namely by applying the rule "the larger the maximum common subgraph, the more similar the graphs". Obviously, the minimum common supergraph can also be used as a measure on graphs (Bunke and Shearer, 1998), or both can be combined into a single measure (Fernández and Valiente, 2001). Moreover, the maximum common subgraph can be used to construct a partial alignment since the partial one-to-one mapping can be directly derived from the maximum common subgraph. A variety of algorithms have been proposed for the calculation of the MCS, some which are exact algorithms using clique-detection (Bron and Kerbosch, 1973; Pelillo, 1998) and, to a lesser extent, also backtracking algorithms (McGregor, 1982; Schmidt and Druffel, 1976). Other approaches approximate the MCS, often based on combinatorial optimization techniques or genetic algorithms (Wagner and Fischer, 1974; Raymond et al., 2002) as the problem is provably NP-hard, which in fact is a major problem for all these methods. Approaches originating in the database field aim at the exploration of (potentially very large) graph databases (Shasha et al., 2002; Yan et al., 2004; Zhang et al., 2007) thus must satisfy a certain degree of efficiency. To reach that goal, these methods use indexing techniques.

A major disadvantage of graph isomorphism is the requirement of exact and complete graph matching that is often not fulfilled in real-world applications. Although subgraph-isomorphism is computationally much more expensive, it has been successfully applied to many problems, in particular to the comparison of protein binding sites (Schmitt et al., 2002). However, due

to its complexity, only small inputs can be handled. Moreover, subgraph isomorphism is still inflexible in the sense that the subgraphs must match exactly, which often leads to small matches in practice. Hence, relaxations e.g. based on quasi-cliques (Liu and Wong, 2008) should be more appropriate for data that is subject to noise and mutations. Unfortunately, the quasi-clique algorithm comes with even higher runtime and memory requirement. A very promising approach, already applied for interaction graph mining, is based on local clique merging (Li et al., 2005). This approach combines both, error-tolerance and efficiency and seems to be a good alternative to the established clique approaches. Due to its potential, the clique merging method will be considered in the second part of this thesis. Other approximate graph matching techniques are given in (Christmas et al., 1995; Suganthan et al., 1995; Xu and Oja, 1990; Wang et al., 1997), which use ideas from the field of computational intelligence. However, these methods are likely to get stuck in local optima.

3.2.2 Methods based on Frequent Subgraph Mining

As pointed out already, frequent subgraph mining aims at identifying a large set of smaller common substructures instead of concentrating on single large subgraph to define similarity on graphs, while offering the opportunity to incorporate multiple graphs into the analysis. Hence, the primary goal of frequent subgraph mining is not similarity analysis. Instead, it is used to detect, e.g., functional groups. Early contributions in this area employ computationally expensive inductive logic programming (Dehaspe et al., 1998; Srinivasan et al., 1997). As this is unfeasible for larger or a greater number of graphs, approximate algorithms have also been proposed (Yoshida and Motoda, 1995; Holder et al., 1994), but these early approaches cannot guarantee to find all common substructures. More advanced methods extend the well-known apriori algorithm (Agrawal and Srikant, 1994) for mining frequent item sets to this problem (Inokuchi et al., 2000; Kuramochi and Karypis, 2001, 2007).

Faster approaches have also been proposed. Borgelt and Berthold (2002); Borgelt et al. (2005) developed an algorithm that employs a depth-first tree search with structural pruning. ClosedGraph is an approach that constraints itself by looking for connected closed subgraphs (Yan and Han, 2003) and FFSM utilizes efficient subgraph enumeration operations (Huan et al., 2003). However, they all constrain the patterns they allow to connected subgraphs. Al-

though these approaches were successfully employed in chemoinformatics, they are generally not applicable for larger graph structures that may arise when analyzing protein structure data due to their complexity.

3.2.3 Methods based on Graph Edit Distance

The methods mentioned above are in most cases dependent on exact matches, although approximations have also been considered to a certain degree. However, as especially in life sciences one has to deal with inconsistencies and noisy data, more powerful approximate and error-tolerant graph matching techniques are required. A powerful alternative to subgraph isomorphism is given by the concept of graph edit distance as a distance measure between graphs, originally introduced by Sanfeliu and Fu (1983). The distance between two graphs is given by the minimal sequence of edit operations needed to transform one graph into the other. Edit operations are typically insertions, deletions and label / weight changes of nodes or edges. This is a more general and more flexible approach to graph matching than the subgraph methods mentioned above. In fact, it could be shown that graph and subgraph isomorphism are special instances of graph edit computations (Bunke, 1999). Therefore exact algorithms cannot solve the graph edit distance problem efficiently (Neuhaus and Bunke, 2007b). Standard methods to compute the graph edit distance are based on search tree algorithms (Tsai and Fu, 1979; Bunke and Allermann, 1983) in which the A* search algorithm (Hart et al., 1968) is used for tree traversal. More recent algorithms that make use of the graph edit distance stem from the field of computer vision. Here, graph edit distances were used in combination with enumeration techniques and indexing methods (Messmer and Bunke, 1998b,a), probabilistic edit models (Myers et al., 2000; Robles-Kelly and Hancock, 2005; Bergamini et al., 2000) or hill climbing heuristics (Wang et al., 1994a,b). Neuhaus and Bunke (2007b) consider fuzzy edit paths which can be modeled as quadratic optimization problem that can be solved by quadratic programming (Nocedal and Wright, 2000). Due to an enormous number of constraints, this approach becomes inefficient even for medium-sized graphs. Another approach uses binary linear programming to calculate graph matchings based on graph edit distance (Justice and Hero, 2006). For special cases of graphs the complexity becomes polynomial. Zhang et al. (1995) introduced an efficient algorithm for graphs with degree 2, Brille (2005) gives a review on

methods for tree alignment. For these types of graph, approaches often make use of techniques from the field of sequence alignment. For graphs which use node labels from $\mathbb{R}^2$, neighborhood graphs can be drawn and transformed into strings afterwards. Subsequent cyclic string matching algorithms (Bunke and Bühler, 1993; Lladós et al., 2001; Peris and Marzal, 2002; Mollineda et al., 2002) can be used to minimize the edit distance on strings (Wagner and Fischer, 1974; Levenshtein, 1966). Such an approach is obviously not exact and returns an approximation of the graph edit distance, however in polynomial time. Just as for exact graph matching, query algorithms for the approximate matching of graphs have been developed as well in the database field (Yan et al., 2005, 2006). Yet, these approaches are still not very flexible, as they do not allow insertions or deletions of nodes. SAGA is a more versatile approach that uses a flexible graph similarity model (Tian et al., 2007). Although SAGA is very efficient on small graphs, it is computationally expensive for large graphs. The proposed TALE algorithm instead allows for the matching of even large graphs by using a novel sophisticated indexing method (Tian and Patel, 2008).

A method which combines subgraph isomorphism and a greedy heuristic to determine graph edit distance on general graphs was presented by Weskamp et al. (2007). This approach is very interesting since the common subgraph is mapped optimally, independent of the used cost function. Generally, this cost function, or more specifically the costs assigned to a certain edit operation, strongly influences the edit distance (Bunke, 1999). Finding an appropriate parameterization for the graph edit distance is often difficult and gives rise to another optimization problem (Neuhaus and Bunke, 2004, 2007a, 2005), which slows down the whole approach. Due to its enormous flexibility, the graph edit distance is a very interesting concept to measure similarity between protein binding sites, which however suffers from its NP-hardness. Therefore, a trade-off must be found between efficiency and exactness. The graph edit distance moreover allows one to derive a one-to-one correspondence between nodes[1], thus it defines a pairwise alignment that can even be extended to a multiple alignment by using standard merging techniques, e.g. based on stars (Altschul and Lipman, 1989), as already applied by Weskamp et al. (2007). Due to these reasons, the graph edit distance will be considered in the following of this thesis in its most appropriate implementation, namely the implementation

[1] Correspondences between nodes allow to derive indirectly correspondences between edges.

of Weskamp et al. (2007).

3.2.4 Methods based on Features and Graph Invariants

Methods that do not consider the graph completely, but instead features or invariants, are interesting since the comparison of vectors can be performed efficiently. The problem here is that graphs of different size must be mapped onto vectors of equal size, moreover, the construction of the vectors can become inefficient, too. The most efficient approaches are based on histograms. Papadopoulos and Manolopoulos (1999) consider the degree of a node and assigns it accordingly to a bin. Another more complex approach captures all information provided by node-labeled and edge-weighted graphs in a set of histograms (Sander et al., 2007).

Instead of using histograms and counting observations, one can define a set of features. Of course, many possibilities of defining features on graphs exist. The main challenge here is to cover all aspects that might be relevant for the underlying problem. One approach to feature definition is to look at local features in a graph in contrast to approaches such as graph edit distance or graph isomorphism, which consider the graph as a whole. Such an approach leads to similarity measures based on local rather than global similarity. Local approaches to graph comparison generally look for the compliance of properties that refer to substructures or local components of a graph, such as subgraphs, paths or walks. In contrast to subgraph isomorphism approaches, local methods typically aim at the identification of a set of characteristic substructures for a given group of graphs rather than the calculation of a single maximum common subgraph.

Main contributions to such similarity measures have recently been made in the field of kernel-based machine learning (Shawe-Taylor and Cristianini, 2003). A kernel function defined on a set $\mathcal{X}$ is an $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ mapping satisfying certain formal properties, including symmetry and positive semi-definiteness, which makes them appealing both from a mathematical and algorithmic point of view. Generally, a kernel function can also be viewed as a similarity function. Several kernel functions on graphs have already been proposed, some of which are based on walks or, more precisely, random walks. Here, walks are generated in one graph at random and then searched in the second graph. The number of random walks present in both graphs can be

used to define a similarity measure on graphs. This can be done in an indirect way as in (Gärtner, 2003, 2008) where properties of the product graph are exploited, or in the direct way in which random walks are drawn and compared afterwards like done by Kashima et al. (2003). Other kernels are closely related to this concept, for example diffusion kernels (Kondor and Lafferty, 2002). Since the number of possible random walks can become extraordinarily large, the use of paths (Borgwardt et al., 2005) and shortest paths (Borgwardt, 2007) has been proposed as an alternative. Graph edit distance can also serve as a tool to define kernels (Neuhaus and Bunke, 2006). A number of kernels exist that are deliberately tailored towards chemoinformatics, namely the Tanimoto kernel, the min-max kernel and the hybrid kernel (Ralaivola et al., 2005). In (Kuhn, 2004) a set of subgraphs of size 3 was considered. For each subgraph, its area and perimeter is determined using as index to generate feature vectors which are subsequently compared using the Tanimoto index.

An interesting approach was suggested by Neuhaus and Bunke (2006). The authors stated that graph kernels and edit distance based matching algorithms tackle the problem of graph similarity in complementary ways, and for given applications, either the first or the second approach is superior to the other. They combine both principles by enhancing a random walk kernel through the addition of information based on graph edit distances. Borgwardt (2007) introduced the so-called graphlet kernel that also makes use of substructures that are subgraphs consisting of four nodes to calculate the similarity between two graphs. The concept of an optimal assignment kernel has been introduced in (Fröhlich et al., 2005). Here, the idea is to search for an assignment of subcomponents of the graphs so that, for a given kernel function on the subcomponents, the sum over all mutually assigned pairs becomes maximal. Strictly speaking, the term “kernel” is misleading here, since this measure does not actually fulfill the kernel properties (Vert, 2008).

Aside from kernel functions, there exist alternative approaches that build upon different feature representations of graphs. One line of works is focused on graph decomposition methods. As every graph can be represented by its adjacency matrix, a number of decomposition methods have been employed to solve the mapping problem of graphs. Several approaches utilize an eigenvalue decomposition of the adjacency matrix (Umeyama, 1988). Kondor and Borgwardt (2008) introduced a set of invariant matrices derived from graphs by the Fourier transformation called the skew spectrum. A comprehensive

summary on graph kernels is given by Vishwanathan et al. (2008).

3.3 Specialized Methods for Protein Structure and Fold Comparison

The methods presented so far were general approaches for the analysis and comparison of general graphs or point clouds which often entail polynomial runtime of high order for the latter case and even exponential runtime for the former case. The more challenging task of a comparative analysis of protein structures, which can have a considerably larger size, renders many of the approaches mentioned above useless for this task. Thus, a variety of specialized approaches exists that aim at the identification of common three-dimensional patterns and substructures in proteins, corresponding to relevant sites for the protein function, such as catalytic triads or protein binding sites. Some of those approaches resort to principles and methods from graph theory. Among these are the ASSAM algorithm (Artymiuk et al., 1994; Spriggs et al., 2003), a method that exploits a protein surface database (ef-site) (Kinoshita and Nakamura, 2003, 2005), and the approach of Jambon et al. (2003), in which the amino acid structure of a protein is represented as a set of chemical groups. These approaches mostly utilize clique detection algorithms to discover similar substructures in graphs. Secondary structure elements, such as alpha helices and beta sheets are used in conjunction with additional information such as the orientation of these elements, to tackle the problem of aligning similar protein structures by employing clique-detection algorithms (Grindley et al., 1993; Mitchell et al., 1990). Based upon this idea, several other algorithms have evolved that employ this strategy (Madej et al., 1995; Alexandrov and Fischer, 1996). In a recent approach, Mernberger et al. (2011) follow a divide-and-conquer strategy in which a graph is decomposed into subgraphs of size three that are processed further. In merging them, additional information provided by protein binding sites is employed to improve the alignment. In (Mizuguchi and Go, 1995), spatial arrangements of secondary structural elements in proteins are compared. The authors in (Shatsky et al., 2004) make use of the C_α atoms, derive a sequence and use this sequence to align a set of labeled point clouds in a very efficient way. Dror et al. (2003) use approximate one-to-one point matching, Leibowitz et al. (1999, 2001) geometric-hashing of secondary

structure elements (SSE) to find multiple structural alignments. Yet other approaches which employ the higher level representation of SSE make use of dynamic programming, depth-first search, three-dimensional clustering or a Markov transition model to align similar SSE (Singh and Brutlag, 1997; Gibrat et al., 1996; Kleywegt and Jones, 1997; Mizuguchi and Go, 1995; Vriend and Sander, 1991; Kawabata and Nishikawa, 2000). Since there are only small numbers of SSE present in a protein structure, algorithms that focus on this representation are usually faster than those building upon a more detailed representation. The concept of SSE also gives rise to a variety of graph based methods, as was mentioned above. However, these methods resemble a fold-based similarity, as SSE are rather coarse features that are not able to capture finer details.

The structural comparison of proteins and protein substructures is not limited to graph- and point cloud-based approaches alone. Approaches beyond these two fields often focus on an alternative protein representation. A general goal of these algorithms is often to derive a suitable sequence alignment backed up by structure information, such as the Euclidean distance between C_α atoms that can be used to superimpose structures. Such alignments are often used to generate a superposition of structures that can be evaluated by the root mean square deviation (rmsd), which is basically a measure for the structural overlap of two superimposed structures. It can be easily calculated by the already introduced Kabsch (1976) algorithm and is often used as a quality measure for structural superpositions. The DALI method, for instance, uses distance matrices of inter-residue distances based on the corresponding C_α atoms to represent proteins and calculates an alignment of structural equivalent residues by using a Monte Carlo approach (Holm and Sander, 1993; Holm and Park, 2000). Other approaches (SSAP, MUTAL) compare inter-atomic distance vectors for a certain residue (Orengo and Taylor, 1996; Taylor et al., 1994) or focus on pairwise residue distances (Gerstein and Levitt, 1996, 1998). MUTAL even allows for the comparison of multiple structures. Shindyalov and Bourne (1998, 2001) proposed an approach based on a combinatorial extension technique (CE). The main drawback of these methods is their relatively large computational cost which renders them less suitable for large-scale analyses. Other approaches aim to find the best superposition of two proteins by minimizing the surface between virtual protein backbones (Falicov and Cohen, 1996). The metric of choice for this task is again the rmsd-value.

Part II

Methods

4

Geometric Approaches

Modeling protein binding sites in terms of a geometric representation has the benefit of a more compact representation and no loss of information since one can work directly with the data provided by CavBase without transforming it to another representation. Here, the so-called *labeled point clouds* are employed which are, e.g., directly used in CavBase to represent protein binding sites. For the comparison of such point clouds, the concept of *labeled point cloud superposition* is introduced. This approach measures similarity between labeled point clouds and moreover returns important additional information, as the multidimensional transformation, that is, a translation and rotation of the second point cloud (as a whole) in the Euclidean space, which superimposes both point clouds optimally with respect to a given fitness function.

As already mentioned, a raw similarity value is sometimes not sufficient, since one seeks to explain the similarity degree or extract additional information, such as conserved patterns. Therefore, with *multiple point cloud alignment* the geometric counterpart to sequence alignment is introduced, which reuses the calculated multidimensional transformation and uses it to calculate the alignment in an optimal and efficient way by solving the well-known problem of bi-partite graph matching. Both approaches, parts of which that were already published in (Fober and Hüllermeier, 2009a), (Fober et al., 2009a), (Fober and Hüllermeier, 2009b) and (Fober et al., 2011) will be introduced in this section beginning with labeled point cloud superposition.

4.1 Labeled Point Cloud Superposition

The similarity between two labeled point clouds is measured in terms of the quality of the best spatial superposition of these two point clouds. The underlying idea is not to tackle the problem on a combinatorial level as, e.g., in graph alignment (Weskamp et al., 2009) but instead to find a problem formulation that allows a more efficient solution. Therefore, point clouds are considered as a whole, i.e., without allowing a change of relative coordinates (inter-point distances) and labels. While holding the first point cloud fix, the second can be moved in terms of a transformation as a whole realized by translation and rotation of all points of the second cloud. However, this transformation only changes the position of the point cloud in the coordinate system, not the point cloud itself.

The algorithmic realization of this concept is based on three steps: First, a fitness function is defined measuring the quality of a certain superposition. This fitness function is used by an optimizer to find the superposition which leads to the maximum fitness. To generate certain superpositions, an appropriate transformation function is defined allowing all possible transformations needed in the 3-dimensional Euclidean space. Having applied a solver on the optimization problem, some information is obtained. Particularly the fitness value reached and the corresponding transformation are of importance, since they express, respectively, the similarity between the point clouds, and the transformation needed to superimpose them optimally.

4.1.1 Quality of a Superposition

To measure the quality of a superposition, an appropriate fitness function must be defined. This section is dedicated to this problem. Let

$$P = \Big\{ \big(x_1, \ell(x_1)\big), \ldots, \big(x_n, \ell(x_n)\big) \Big\}$$

be a point cloud consisting of n points $x_i = (x_{i1}, x_{i2}, x_{i3}) \in \mathbb{R}^3$ with associated label $\ell(x_i) \in \mathcal{L}$, where $\mathcal{L}$ is a discrete set of labels. In this thesis, $\mathcal{L}$ is given by the seven types of physicochemical properties. Moreover, let

$$P' = \Big\{ \big(y_1, \ell(y_1)\big), \ldots, \big(y_{n'}, \ell(y_{n'})\big) \Big\}$$

be a second point cloud to be compared with P, where P' consists of n' points $y_i = (y_{i1}, y_{i2}, y_{i3}) \in \mathbb{R}^3$ with associated label $\ell(y_i) \in \mathcal{L}$. The origin of the point

clouds in a coordinate system is completely arbitrary, since the coordinates were determined experimentally without ensuring a common coordinate system. However, the quality of the given superposition $\mathbf{0}$ (which will be very poor) can be calculated here by use of a generalized fuzzy equivalence (cf. Section 2.4.1), where for ordinary sets A and A', the equivalence can be reduced to two inclusions. Here, the calculation of the degree $A \subseteq A'$ can be realized in at least two ways: By a straightforward generalization of the inclusion and alternatively by using a generalization of the well-known Jaccard coefficient.

Generalization based on the Inclusion for Sets

For sets A and A', the subset relation $A \subseteq A'$ is calculated by determining if each point $a \in A'$ is also present in A. Hence, for those sets the inclusion $A \subseteq A'$ is defined as $\forall\ a \in A' \Rightarrow a \in A$. Therefore, to adapt this concept to point clouds, first the question must be answered whether a point $y \in P'$ is also present in P. More specifically, the degree to which $y \in P'$ is at least "fuzzily" present in P must be determined. In Section 2.2, it was shown that a fuzzy subset F of a reference set U is characterized by its membership function, which is a $U \to [0,1]$ mapping μ_F which generalizes the characteristic function of a set (Zadeh, 1983). For each $u \in U$, $\mu_F(u)$ is the degree of membership of u in the fuzzy set F.

For a fixed $y \in P'$, the membership degree of this point in P, that is, the degree to which this point is also present in P, is defined by

$$\mu_P(y) = e^{-\gamma \cdot d(y,P)} \ , \tag{4.1}$$

where

$$d(y,P) = \min_{\substack{x \in P \\ \ell(x)=\ell(y)}} \|y - x\|_1$$

is the distance between a point $y \in P'$ and the closest point $x \in P$ having the same label measured in terms of the L_1 distance. If such a point does not exist, $d(y,P) = \infty$ is used and hence $\mu_P(y) = 0$. For points $x \in P$, the membership degree $\mu_{P'}(x)$ and the distance $d(x,P')$ are defined analogously.

Having defined a function calculating the degree for the presence of a point y in a point cloud P allows one to go one step further, thus to adapt the crisp inclusion on sets $\forall\ a \in A' \Rightarrow a \in A$ to point clouds. For fuzzy sets, analogous concepts exist to model all-quantifier and inclusion. Often, the universal-quantifier is substituted by the infinum and the inclusion operator by a fuzzy

realization of the inclusion (cf. Section 2.2). Thus, one obtains for labeled point clouds P and P'

$$P' \subseteq P \Leftrightarrow \inf_{y \in P'} I(1, \mu_P(y)) = \inf_{y \in P'} \mu_P(y) \,, \tag{4.2}$$

where $I : [0,1]^2 \to [0,1]$ is the fuzzy realization of the inclusion operator for which $I(1,x) = x$ is an axiom. Unfortunately, the infimum operator in equation (4.2) leads to undesired effects when working on data that are noisy and subject to mutations and structural flexibility since it is sensitive to that point that leads to the infimum. Thus, if even almost all points match perfectly, the outlier would dominate the measure (4.2), hence a small inclusion value would be obtained, leading to an overall small similarity value. Therefore, to relax this definition, the infimum is replaced by a fuzzy quantifier Q, which is specified in the form of a non-decreasing $[0,1] \to [0,1]$ mapping (Fodor and Yager, 2002; Zadeh, 1983). This leads to

$$\min_{i=1,\ldots,|P'|} \max\{Q(i/|P'|), m_i\} \,, \tag{4.3}$$

where m_i is the i-th largest membership degree in the fuzzy set $\{\mu_P(y) \mid y \in P'\}$. On the one hand, (4.3) is a generalization of (4.2), since (4.2) can be obtained by defining $Q(1) = 1$ and $Q(t) = 0$ for $0 \leq t < 1$ in (4.3). Moreover it is a very intuitive measure for assessing the quality of a superposition of two point clouds that is based on the inclusion of ordinary sets. On the other hand, however, (4.3) is not the optimal choice since it does not fulfill continuity and exhibits many plateaus[1], which makes calculations very difficult. Therefore a further measure is defined based on the Jaccard coefficient.

Generalization based on the Jaccard Coefficient

To overcome the problem of plateaus and discontinuity, another property of the inclusion can be used: For a pair of sets P and P', the inclusion $P' \subseteq P$ is equivalent to the equality $P' = P' \cap P$. Consequently, one possibility for relaxing the inclusion relation on sets to a fuzzy inclusion is to make use of a corresponding fuzzy equivalence such as, e.g., the Jaccard measure $|X \cap X'| / |X \cup X'|$ on two sets X and X'. In the fuzzy case, set intersection and union are accomplished

[1] Discontinuity is violated if the value the maximum operator returns changes from the first to the second argument. Plateaus occur because for a number of transformations the value (4.3) returns is unchanged.

through t-norm and t-conorm operators $\top$ and $\bot$, respectively, and set cardinality through summing membership degrees. Noting that $\mu_{P'}(y) = 1$ for all $y \in P'$ and that 1 is the neutral element of a t-norm $\top$, this eventually yields

$$\begin{aligned} inc(P', P) &= \frac{\left|P' \cap (P' \cap P)\right|}{\left|P' \cup (P' \cap P)\right|} \\ &= \frac{\left|P' \cap P\right|}{\left|P'\right|} \\ &= \frac{\sum_{y \in P'} \top(\mu_P(y), \mu_{P'}(y))}{\sum_{y \in P'} \mu_{P'}(y)} \\ &= \frac{1}{|P'|} \sum_{y \in P'} \mu_P(y) \,. \end{aligned} \tag{4.4}$$

Hence, (4.4) also allows measurement of the quality of two superimposed point clouds measured in terms of a generalized inclusion on sets and comes with continuity and without plateaus, two very important properties, as will be shown later.

4.1.2 Transformation of a Point Cloud

As mentioned above, the idea of this approach is to define the similarity between two labeled point clouds in terms of the quality of the best superposition of these two clouds. To this end, three steps were defined, the first of which is solved so far. In the following, the next sub-problem is tackled, namely the transformation of a point cloud. As mentioned above, the flexibility of point clouds is restricted to enable an efficient similarity measure. Here, a restricted flexibility means that point clouds are not allowed to change the labels or the coordinates of certain points. While the change of a label is strictly forbidden, the coordinates of the point cloud can be changed, however, as a whole. This is realized by a function TF : $\mathcal{P} \times [0, 2\pi] \times \mathbb{R}^3 \to \mathcal{P}$ that moves a point cloud via rotation and translation, as specified by a six-dimensional vector $t = (\theta_1, \theta_2, \theta_3, \delta_1, \delta_2, \delta_3) \in [0, 2\pi]^3 \times \mathbb{R}^3$.

Concretely, the transformation is composed by a rotation followed by a translation. To apply rotation, the point cloud is shifted to the origin of the coordinate system. This is done by determining the center of gravity

$$p_C = \frac{1}{n} \cdot \sum_{i=1}^{n} p_i \in \mathbb{R}^3 \,,$$

of the point cloud $P = \{(p_1, \ell(p_1)), \ldots, (p_n, \ell(p_n))\}$, and by adding $-p_C$ to all points $p_i, i = 1, \ldots, n$. This preprocessing step allows one to rotate the point cloud sequentially on x-, y- and z-axes according to equation (4.5), (4.6) and (4.7), respectively.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \cdot \cos(\theta_1) + z \cdot \sin(\theta_1) \\ -y \cdot \sin(\theta_1) + z \cdot \cos(\theta_1) \end{pmatrix} \tag{4.5}$$

$$\begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix} = \begin{pmatrix} x' \cdot \cos(\theta_2) - z' \cdot \sin(\theta_2) \\ y' \\ x' \cdot \sin(\theta_2) + z' \cdot \cos(\theta_2) \end{pmatrix} \tag{4.6}$$

$$\begin{pmatrix} x''' \\ y''' \\ z''' \end{pmatrix} = \begin{pmatrix} x'' \cdot \cos(\theta_3) + y'' \cdot \sin(\theta_3) \\ -x'' \cdot \sin(\theta_3) + y'' \cdot \cos(\theta_3) \\ z'' \end{pmatrix} \tag{4.7}$$

To recover the original position of the point cloud, i.e., of the center of gravity, the translation is undone after rotation by adding p_C to each point of the point cloud. Finally, to complete the transformation, translation is applied. The translation to perform is given by the vector $\delta = (\delta_1, \delta_2, \delta_3)$ and realized by

$$p_i = p_i + \delta\,, \quad i = 1, \ldots, n$$

for point clouds of size n. To accelerate the calculation, the last step of the rotation, namely the translation to the original center of gravity and the translation by the vector δ should be combined.

Note that all these operations leave the label information unchanged, i.e., $\ell(p_i)$ before transformation is equal to $\ell(p_i)$ after transformation.

4.1.3 Optimizing the Superposition

The function $\mathrm{TF}(\cdot, t)$ allows one to move a point cloud via rotation and translation, as specified by the six-dimensional vector $t = (\theta_1, \theta_2, \theta_3, \delta_1, \delta_2, \delta_3) \in [0, 2\pi]^3 \times \mathbb{R}^3$. Thus,

$$P^* = \mathrm{TF}(P, t) = \{(y_1^*, \ell(y_1^*)), \ldots, (y_n^*, \ell(y_n^*))\}$$

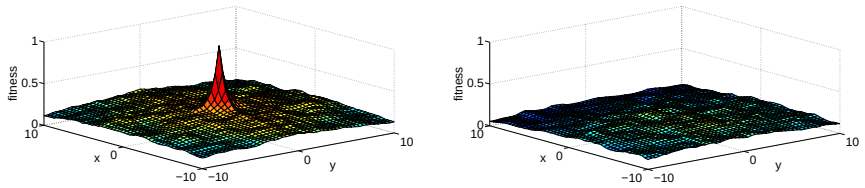
is the transformed point cloud obtained by rotating the point cloud P by the angles $\theta = (\theta_1, \theta_2, \theta_3)$ and translating the obtained result by $\delta = (\delta_1, \delta_2, \delta_3)$. The function $\mathrm{inc}(\cdot,\cdot)$, furthermore allows evaluation of the superposition of two point clouds. Obviously, due to the experimental determination of the shape of a protein, protein binding sites stored in CavBase are not superimposed (optimally). Therefore, in the last step of the labeled point cloud superposition approach, both functions will be combined to enable a search for the optimal superposition and hence for the position-invariant degree of inclusion of P' in P, that is then given by

$$\mathrm{INC}(P', P) = \max_{t \in [0,2\pi]^3 \times \mathbb{R}^3} inc(\mathrm{TF}(P', t), P) \ . \tag{4.8}$$

The degree of inclusion of P in P', $\mathrm{INC}(P, P')$, is given analogously.

Solving the Optimization Problem

The computation of the inclusion (4.8) involves the solution of a real-valued optimization problem, namely the problem of finding an optimal vector t and, thus, an optimal point cloud superposition. The objective function to be maximized here is highly non-linear and multimodal. As an illustration, Figure 4.1 (a) shows the objective function obtained for the superposition of a randomly generated two-dimensional point cloud P in which all points have the same label. This function maps each two-dimensional translation vector $t = (x, y)$ to the corresponding similarity degree between $\mathrm{TF}(P)$ and P, where no rotation was considered. As can be seen, there is a sharp peak at $t = (0, 0)$, which corresponds to the optimal superposition. Surrounding this solution, however, there are also many local optima. Figure 4.1(b) shows an analogous example in which two different, randomly generated point sets are used. Here the optimum is not in $t = (0, 0)$ and a peak does not stick out as expected for completely dissimilar point clouds. The problem of local optima also becomes clear from this small example: Moving the point cloud P from left to right, into the direction of P', has the following effect: First, a good superposition of two subclouds will be found, namely the right part of cloud P and the left part of cloud P'. This results in a local maximum. Moving P further to the right leads to a larger local maximum (sub-clouds are growing), until the global maximum is eventually reached. To solve the optimization problem, an *evolutionary strategy* (ES), a population-based, stochastic optimization method inspired by biologi-



(a) Randomly generated point cloud and its identical copy

(b) Two randomly generated point clouds

Figure 4.1: Objective functions of LPCS in the case of (a) two identical point clouds and (b) two randomly generated point clouds.

cal evolution and specifically developed for real-valued optimization problems (Beyer and Schwefel, 2002) is used (cf. Section 2.3).

Population-based optimization methods are especially advantageous for highly multimodal problems. Using a large population leads to an increased probability of generating a candidate solution in a region where the direction of ascent points to the global maximum. Especially direct search approaches, such as evolutionary strategies, make no demands on the function they optimize. Only the ability to evaluate each point of the search space is required, moreover, having in each point a direction of decent leads as a rule to a much higher convergence velocity. Therefore, the membership function (4.1) is chosen here as the function realizing the membership degree since it is a strictly monotone decreasing function which converges to zero and ensures to have the direction of descent in each point $t \in [0, 2\pi]^3 \times \mathbb{R}^3$. This greatly simplifies the detection of the solution of the maximization problem.

Complexity

Even though evolution strategies are generally known to be quite efficient solvers, the concrete complexity does of course depend on the application at hand. The application-specific part is the fitness function, i.e., the objective function to be optimized. This function has to be evaluated frequently and, therefore, is an important factor for the runtime. In the case considered here, this function is given by the similarity measure (4.8), and its evaluation is strongly dominated by the nearest neighbor search which has to be conducted for each single point in both structures, since according to (4.1), membership degrees are determined by the distance to closest points with the same label.

There exist many data structures for supporting nearest neighbor search, see e.g. (de Berg et al., 2000) for a comprehensive introduction. The most efficient among them need time $\mathcal{O}(n \log^2 n)$ for construction and $\mathcal{O}(\log^3 n)$ for answering a query. Compared to a naive approach which needs time $\mathcal{O}(n)$ a considerable speed-up can be reached. However, data structures always come with certain overhead that dominates the whole approach if the size of the data is very small. In the application considered here, protein binding sites are processed which are characterized by 91.62 points on average (even though much larger structures do of course exist). Therefore the use of a complex data structure does not pay off. Nevertheless, efficiency can be increased by hashing the points x_i of a point cloud, using the label $\ell(x_i) \in \mathcal{L}$ as a key. Since nearest neighbors are only sought among points with the same label, this obviously reduces the size by a factor of approximately $|\mathcal{L}|$. Hence, the usage of a "naive" approach which is considering each point and returns the closest, is the most efficient one: It is coming with complexity $\mathcal{O}(n)$. Hence an overall fitness of $\mathcal{O}(n^2)$ is obtained for one fitness-evaluation.

Even though the complexity of the fitness evaluation is known, the overall complexity of the LPCS approach cannot be determined. Evolutionary algorithms are complex randomized algorithms for which an analysis of complexity is not easy to perform. There are only few examples of problems for which an analysis could be performed successfully (Droste et al., 2002), by using assumptions that are not applicable for real-world applications. Without fixing the number of iterations that the evolutionary algorithm performs, such calculations are not possible here. Therefore, the runtime is expressed by utilizing the random variable l, giving the function evaluations required to fulfill a termination criterion. The complexity of the whole approach is hence given by $l \cdot \mathcal{O}(n^2)$, where $n = \max\{|P|, |P'|\}$.

Since the point clouds are considered directly and calculations are performed on vectors of size 6, the space complexity of the overall method is linear in the size of the point clouds.

4.1.4 Defining Similarity

Based on the calculated degrees $INC(P, P')$ and $INC(P', P)$, the similarity between P and P', in the sense of a generalized equivalence, can be defined as

$$\mathrm{SIM}(P, P') \;=\; \min\{\, \mathrm{INC}(P, P'),\, \mathrm{INC}(P, P') \,\} \;. \tag{4.9}$$

To reach more flexibility, however, the two inclusion degrees can be combined as introduced in Section 2.4.1, to find a trade-off between the two extreme aggregation modes min (used in (4.9)) and max. The similarity measure adjustable by the parameter λ thus becomes

$$\begin{aligned} \mathrm{SIM}(P,P') =& \lambda \cdot \min\{\mathrm{INC}(P,P'),\mathrm{INC}(P',P)\} \\ &+ (1-\lambda)\cdot \max\{\mathrm{INC}(P,P'),\mathrm{INC}(P',P)\}\ . \end{aligned} \tag{4.10}$$

4.2 Alignment of Labeled Point Clouds

With labeled point cloud superposition, a novel method was developed to measure similarity between geometric data. For CavBase data, this has the enormous advantage that a further transformation of protein binding sites becomes unnecessary, a process often connected with a loss of information. As already mentioned, however, often a raw similarity is not sufficient, especially if one wants to explain the obtained similarity or to analyze a whole set of protein binding sites. In such a case the detection of common substructures or conserved patterns is required. This problem was so far only tackled for objects represented by graphs, where, e.g., the concept of multiple graph alignment (Weskamp et al., 2007) was introduced for this purpose. Here, a similar concept is developed which can however be applied directly on geometric data, namely labeled point clouds. As already in the case of graph alignment, the goal of labeled point cloud alignment is to establish a one-to-one correspondence between the basic constituents of the structures, namely labeled points.

4.2.1 Multiple Point Cloud Alignment

Multiple point cloud alignment (MPCA) is defined according to Definition 2.17, where the set $\{X_i \mid i = 1,\ldots,m\} = \mathcal{X}$ becomes $\{P_i \mid i = 1,\ldots,m\} = \mathcal{P}$. Hence, one is looking for a one-to-one correspondence between the points of different point clouds. The number of valid alignments is enormous, as already shown. This makes it necessary to search for the best alignment out of the set of valid alignments, that is an alignment which reflects structural correspondence in an optimal way. In the case of graph alignment, Weskamp et al. (2007) solved this problem by defining a scoring function on the combinatorial search space which was optimized by a greedy algorithm. This simple idea could of course also be applied to the multiple point cloud alignment problem.

Unfortunately, combinatorial optimization is in this case NP-hard, leading to the well-known problem that a trade-off must be found between the quality of the solution and the runtime. Therefore another approach is developed here. The idea is to derive an optimal MPCA from an optimal superposition of the labeled point clouds. More specifically, the pairwise alignment is defined on the basis of a given superposition. As will be seen below, the problem of finding an optimal (pairwise) alignment thus comes down to solving several linear assignment problems. For solving the *multiple* point cloud alignment problem, a two step-based approach can be employed, first solving the problem of aligning two structures and secondly merging the pairwise alignments to a multiple alignment. Techniques introduced in Section 2.5 can be used for this purpose. Alternatively, Shatsky et al. (2006) make use of m-partite pivot graph matching which solves the multiple point cloud alignment directly without decomposing it into a set of pairwise problems.

4.2.2 Construction of Pairwise Alignments

The construction of a pairwise alignment of two point clouds P and P' can be reduced to an optimal assignment problem. To this end, a square matrix $M = (m_{i,j})$ is needed, where $m_{i,j} \in \mathbb{R}$ defines the cost for assigning point $p_i \in P$ to point $p_j \in P'$. According to Definition 2.17, the maximal length of a pairwise alignment is $n = |P| + |P'|$. Therefore, to consider all possible alignments, the matrix M has size $n \times n$.

The entries $m_{i,j}$ are derived from the optimal superposition of point clouds P and P' as produced by a modification of the LPCS method. This modification concerns the similarity measure to be maximized. Since a mutually optimal alignment is sought, the similarity is not split into two optimal degrees of inclusion, as in measure (4.10). Instead similarity is defined in terms of a compromise measure as follows:

$$\mathrm{SIM}_{\mathrm{PCA}}(P, P') = \max_{t \in [0,2\pi]^3 \times \mathbb{R}^3} F(P, P', t), \tag{4.11}$$

where

$$F(P, P', t) = \frac{1}{2} \cdot \Big(inc\big(TF(P', t), P\big) + inc\big(P, TF(P', t)\big) \Big) .$$

Given a spatial superposition optimal in the sense of (4.11), it makes sense to define the cost $m_{i,j}$ in terms of the associated L_1 distance d_M between point $p_i \in P$ and $p_j \in P'$. To account for point-to-dummy mappings, the distance between

a point and a dummy is specified by a parameter k. Finally dummy-dummy assignments are scored by zero, so that these mappings will not influence the construction of the alignment. As an illustration, Table 4.1 shows a matrix M for two point clouds $P = \{a, b, c, d\}$ and $P' = \{a', b', c'\}$ and Figure 4.2 the resulting bi-partite graph.

Table 4.1: Matrix representation of the optimal assignment problem.

	a'	b'	c'	$\perp$	$\perp$	$\perp$	$\perp$
a	$d_M(a,a')$	$d_M(a,b')$	$d_M(a,c')$	k	k	k	k
b	$d_M(b,a')$	$d_M(b,b')$	$d_M(b,c')$	k	k	k	k
c	$d_M(c,a')$	$d_M(c,b')$	$d_M(c,c')$	k	k	k	k
d	$d_M(d,a')$	$d_M(d,b')$	$d_M(d,c')$	k	k	k	k
$\perp$	k	k	k	0	0	0	0
$\perp$	k	k	k	0	0	0	0
$\perp$	k	k	k	0	0	0	0

Formally, the assignment problem, also known as the weighted bi-partite matching problem, is specified by a graph $G = (V, E, \ell_E)$ with $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$ and $E = \big\{(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2\big\}$. Moreover, each edge $e \in E$ has an associated cost value $\ell_E(e)$. The goal is to find a subset of edges $M \subseteq E$ solving the following constrained optimization problem:

$$\text{minimize} \sum_{e \in M} \ell_E(e) \tag{4.12}$$

subject to

$$\bigcup_{(v_1,v_2)\in M} \{v_1\} = V_1\,, \qquad \bigcup_{(v_1,v_2)\in M} \{v_2\} = V_2\,, \tag{4.13}$$

and such that $(v_1, v_2) \in M$ and $(v_1', v_2') \in M$ with $(v_1, v_2) \neq (v_1', v_2')$ implies $v_1 \neq v_1'$ and $v_2 \neq v_2'$. In other words, M defines a bijection between V_1 and V_2. In this case, the sets V_1 and V_2 represent, respectively, the points in point cloud P (supplemented with $|P'|$ dummy points) and P' (supplemented with $|P|$ dummy points). Moreover the cost $\ell_E(e)$ of an edge $e = (v_i, v_j)$ is given by the corresponding matrix entry $m_{i,j}$ representing the distance between both points in the optimal superposition according to (4.11). See Figure 4.2 for an illustration.

To solve the weighted bi-partite graph matching problem, the Hungarian algorithm (Kuhn, 2005) is used. Once a cost-minimal assignment has been found, the point cloud alignment is defined by the corresponding point-to-point and point-to-dummy assignments, while all dummy-to-dummy assignments are ignored.

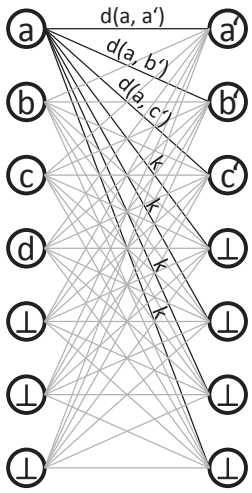


Figure 4.2: Illustration of the weighted bi-partite graph matching problem.

Complexity

The space complexity of this approach is given by the size of the matrix representing the bi-partite graph, since the Hungarian algorithm works directly on this matrix. The size of this matrix is obviously $(|P| + |P'|)^2$. The time complexity of this approach was given by Kuhn (2005), who reported a cubic runtime, hence $\mathcal{O}((|P| + |P'|)^3)$ in the case considered here.

4.2.3 Construction of Multiple Alignments

The ability to construct pairwise alignments obviously allows directly the calculation of multiple alignments, since techniques like star- or tree-alignment (cf. Section 2.5) are able to merge these pairwise alignments to a multiple alignment. Moreover, the concept of *m*-partite graph matching can be applied in a straightforward way by deriving the costs for assignments from the optimal superpositions as done in the pairwise case.

4.3 Summary

As already described in Section 2.1, point clouds as introduced here are far from allowing the flexibility graphs do, especially because the proposed algorithm does only allow to transform a point cloud as a whole. But even if the maximum degree of modification would be allowed, there are only n points available in a point cloud of size n which can be modified by shifting them in the 3-dimensional space. A graph representing a point cloud of size n captures geometry, on the other hand, by storing $n^2/2$ distances which can be measured between pairs points. These distances can be used to modify the graph descriptor. Hence, a much more flexible model is obtained. The restricted flexibility of point clouds, however, is not necessarily a disadvantage: First of all, it enormously simplifies calculations. Moreover, graphs as a representation of relations can be used to describe any kind of data. This often enhances the search space, hence algorithms perform less efficiently.

A geometric representation of a protein binding site is the most compact one, allowing for efficient calculations even on large input-sizes. Here, two different concepts were developed, a measure expressing similarity between two point clouds in terms of an optimal superposition of the point clouds, and furthermore a technique using the calculated superposition to derive one-to-one correspondence between points. Using sophisticated functions and modeling techniques, the underlying problem allows for an efficient solution. The problem defined for determining the optimal superposition becomes a continuous real-valued optimization problem. Even though this problem is multimodal and not continuously differentiable, evolutionary strategies can be used to find the global optimum reliably and quickly. The problem of finding an optimal pairwise alignment was reduced to the bi-partite graph matching problem which can be solved in polynomial time. For constructing multiple alignments, star-alignment, tree-alignment or the technique based on the m-partite pivot graph can be used.

An important advantage of a geometric representation is that the geometric form (e.g. convex or concave) of a patch is available during calculations. Hence, the expensive additional step of superimposing and scoring 100 solutions, which is applied in CavBase, becomes obsolete. This will lead to a dramatical speed-up, in particular, because the scoring is known as the bottle-neck of CavBase.

5

Feature-based Approaches

A simple method to measure similarity between complex objects is to map them to feature vectors, thus reducing the comparison of these complex objects to a comparison of vectors (sometimes also called *fingerprints*). This procedure requires the definition of a set of features, whose elements have an arbitrary but fixed position in the vector. The definition of these features is clearly a crucial step since the effectiveness and the efficiency of the resulting approach clearly depend on them. On the one hand, if a very small set of less informative features is used, the effectiveness will be small. On the other hand, too many features whose detection is possibly computationally complex, result in an inefficient similarity measure. Thus, one has to find a trade-off between both extremes.

To capture the geometry of a protein binding site, the features must consider distances between the pseudocenters. Obviously, an observed distance cannot serve as a feature or part of a feature, since distances are drawn from real numbers and the probability of observing a certain distance is zero. Therefore, instead of considering distances, one has to consider bins of certain length as features or part of them, alternatively distances can be discretized in an appropriate way. Defining bins or discretizing, however, comes with the problem of *discontinuity on bin-boundaries*. Here, methods from the field of fuzzy logic can help to solve this problem in an elegant way.

Having defined the set of features and ordered them, in the next step the occurrence of each feature in a protein binding site is counted resulting in a vector for the considered protein binding site. To compare two protein binding sites now one can make use of the vector representation and a large number of measures on vectors. Even though some realizations of feature-based approaches

will not be very efficient, they come with the advantage that each protein binding site has to be transformed exactly once. The resulting vector can be stored afterwards in addition to the binding site in a database. This would accelerate the calculations enormously since measures on vectors are usually very efficient and the transformation into a feature vector would become obsolete from the second calculation on.

An even simpler approach neglects the definition of features and considers directly the given data. Here, obviously pseudocenters and distances between pseudocenters can be observed, thus at least two sets. Histograms can be used to approximate their distribution. Generally, a histogram is a partition of a set of observations into a finite number of discrete units. Having defined this set of units, again, as in the case of feature vectors, for each unit the number of observations is counted that falls into this unit. As already for the case of feature vectors, there exist a large set of measures on histograms that can be used to reduce the problem of comparing protein binding sites to the problem of comparing histograms. The definition of discrete units, however, comes again with the problem of discontinuity on bin-boundaries that is once more solved by means of fuzzy logic leading to so-called fuzzy histograms.

In this thesis a distinction between *observation* and *feature* is performed. A feature can be an arbitrary “pattern“, whose detection might become computationally expensive (e.g. the detection of a subgraph in a graph). An observation, on the other hand, is directly visible in the data, as the term already indicates. In the following, all variants are presented in more detail beginning with approaches based on histograms that are clearly the simplest way to compare protein binding sites, followed by a more complex method based on features. In this chapter, however, the question as to which of these methods is the best choice for the problem considered here cannot be answered, even though there is much evidence that feature vectors will lead to the best results since they are very flexible and allow specialized features appropriate for representing protein binding sites. In any case, it is of interest how the more efficient histogram-based approaches will perform, and if it is possible to use them as a kind of filter to accelerate calculations. Even if this filter would lead to both a high true-positive rate and a high false-positive rate, it would allow the removal of a large set of protein binding sites that cannot be a hit. Afterwards, on this reduced set more exact and more complex methods can eventually be applied. Altogether, this would lead to an acceleration of cal-

culations especially on such a large database as CavBase. Parts of this chapter were already published; histogram-based representations were published in (Fober and Hüllermeier, 2010); methods based on feature vectors in (Fober et al., 2012).

5.1 Histogram Representations

In this section, the problem of comparing protein binding sites is reduced to the approximation of the distributions of physicochemical properties and distances given in protein binding sites, and the subsequent comparison of these approximations. To approximate a distribution, histograms can be used, hence the representation of protein binding sites can be realized in terms of histograms. For these histograms, a large number of measures exist that can be used for comparison, thus for comparison of protein binding sites that are represented by these histograms.

Formally, a histogram h on a set $\mathcal{X}$ of objects can be represented as a $\mathcal{B} \longrightarrow \mathbb{R}$ mapping, where $\mathcal{B}$ is a finite set of bins, and $h(b)$ denotes the number or fraction of observations $\mathcal{O} \subset \mathcal{X}$ falling into bin b. A histogram h is called normalized if $\sum_{b \in \mathcal{B}} h(b) = 1$. Each bin b is associated with a subset $X[b]$ of the domain $\mathcal{X}$, so that $h(b) = |\mathcal{O} \cap X[b]|$ before normalization and

$$h(b) = \frac{|\mathcal{O} \cap X[b]|}{|\mathcal{O}|}$$

in the normalized case. The set of bins is assumed to form a partition of $\mathcal{X}$, i.e., $X[a] \cap X[b] = \emptyset$ for $a \neq b$ and $\bigcup_{b \in \mathcal{B}} X[b] = \mathcal{X}$. The most common example is a partitioning of the reals, in which case the bins b are associated with intervals $X[b] \subset \mathbb{R}$. In most cases, in particular because a histogram represents a distribution, normalized histograms are used, e.g. Bronstein et al. (2008) use normalization directly in the definition of a histogram and do not consider unnormalized histograms. In the case considered here, a normalization comes moreover with the advantage of a size-invariant measure, i.e., large protein binding sites can reach the same maximal value as small protein binding sites.

In the following, two approaches are presented which derive a set of histograms for a protein binding site, each of them considering certain distributions.

5.1.1 Handling Properties and Distances Separately

A protein binding site was already introduced as a set of pseudocenters $\mathcal{P} = \{p_1, \ldots, p_n\}$ having coordinates in the Euclidean space and labels taken from the set of physicochemical properties. Therefore, for a protein binding site two multisets can be observed, namely the multiset

$$\mathcal{L} = \{\ell(p_1), \ldots, \ell(p_n)\}$$

of physicochemical properties that appear in the protein binding site and derived from $\mathcal{P}$ the multiset

$$\mathcal{D} = \{d_{i,j} = d_E(p_i, p_j) \mid p_i, p_j \in \mathcal{P},\ i < j\}$$

of all pairwise distances between the pseudocenters in $\mathcal{P}$, measured in terms of the Euclidean distance d_E.

Histograms are used to approximate the underlying distribution of both multisets $\mathcal{L}$ and $\mathcal{D}$. To derive the histogram approximating $\mathcal{L}$, the bins $\mathcal{B} = \{1, 2, 3, 4, 5, 6, 7\}$ are used, where each integer is associated with a physicochemical property. Thus, for each type of physicochemical property, simply its relative number of occurrences is counted. For pairwise distances between pseudocenters, the set of bins $\mathcal{B} = \{1, \ldots, d_{\max}\} \subseteq \mathbb{N}$ is used, where $d_{\max}$ is an upper bound on the edge length measured in the unit Å that was defined in a preprocessing step by taking the smallest upper bound valid for the dataset at hand. So, $h(b)$ is the percentage of edges whose length is in $]b, b+1]$.

Complexity

The time needed for construction of both histograms depends on the size of the protein binding site measured in terms of the number of pseudocenters, thus on $n = |\mathcal{P}|$. The construction of the first histogram approximating the distributions of the multiset $\mathcal{L}$ comes with complexity $\mathcal{O}(n)$, since for each physicochemical property an assignment to the correct bin must be performed that takes $\mathcal{O}(1)$. For construction of the second histogram, thus of the multiset $\mathcal{D}$, obviously time $\mathcal{O}(n^2)$ is needed, since $\mathcal{O}(n^2)$ distances are assigned to a bin, where each assignment again needs time $\mathcal{O}(1)$. The space complexity is given by the maximal number of bins which is equal to $d_{\max}$, a generally small number.

5.1.2 Handling Properties and Distances Jointly

Considering distances and physicochemical properties separately leads to a very efficient approach. However, it obviously comes on the other hand with a loss of information since both information are not combined. This might lead to the case that protein binding sites which exhibit a similar geometry and a similar distribution of physicochemical properties are considered as equal, even though the physicochemical properties are spatially completely differently placed. In any case, it is interesting to test if such a simple approach already leads to passable results. Moreover, this approach can be used as a starting point for a more complex method.

A straightforward extension of the first approach, which remains very efficient, is to combine both distances and physicochemical properties. A distance consists of two points, each of which is labeled with one of the seven available physicochemical properties. Therefore, $7 \cdot 8/2 = 28$ multisets $\mathcal{D}_{i,j}$ can be considered, which are defined as follows:

$$\mathcal{D}_{i,j} = \{d_{k,l} = d_E(p_k, p_l) \mid p_k, p_l \in \mathcal{P},\ \ell(p_k) = i,\ \ell(p_l) = j,\ k < l\}\ ,$$

for $1 \leq i \leq j \leq 7$, where each integer corresponds to a physicochemical property, and where again distances are measured in terms of the Euclidean distance. Hence, each multiset $\mathcal{D}_{i,j}$ contains the distances between pseudocenters of type i and j, $(i, j \in \{1, \ldots, 7\})$. Again, for each $\mathcal{D}_{i,j}$, the distribution is approximated in terms of a histogram $h_{i,j}$, for which the bins $\mathcal{B} = \{1, \ldots, d_{\max}\}$ are used. All histograms are normalized so as to give them the same weight, except those histograms that are empty: These histograms remain empty. The resulting histograms are still one-dimensional. However, this type of representation has the advantage of combining information about physicochemical properties and distances, therefore about the chemical properties and the geometry of a protein binding site.

The price to pay is a larger number of features or more precisely, since 28 histograms per protein binding site must be considered, a larger number of comparisons. However, as will be shown later, the theoretical complexity for the construction of all histograms remains the same as for the case above, in particular, because this representation is still a strong simplification using one-dimensional observations to describe a three-dimensional structure. In Section 5.2 therefore a method is proposed that uses simplices as features that are able

to describe the surface of the protein binding site.

Complexity

As already mentioned, the complexity of the method transforming a protein binding site into these 28 histograms is of the same polynomial order as the simpler approach, in which two histograms are considered for each protein binding site. Hence, the time complexity remains $\mathcal{O}(n^2)$, since there are again $\mathcal{O}(n^2)$ distances that can be observed, each of which is assigned to a certain histogram and bin in time $\mathcal{O}(1)$. The space complexity is given by $28 \cdot d_{\max}$, since 28 histograms are used of size $d_{\max}$.

5.1.3 Fuzzy Histograms

Fuzzy histograms are introduced here as a tool that solves the aforementioned problem of discontinuity on bin-boundaries on histograms representing the multisets $\mathcal{D}$ and $\mathcal{D}_{i,j}$. The considered multiset $\mathcal{D}$ or the multisets $\mathcal{D}_{i,j}$, respectively, contain real numbers that are assigned to intervals having crisp interval boundaries. These crisp boundaries cause some problems since they are to some extent arbitrary, and in many cases a small change of a boundary may

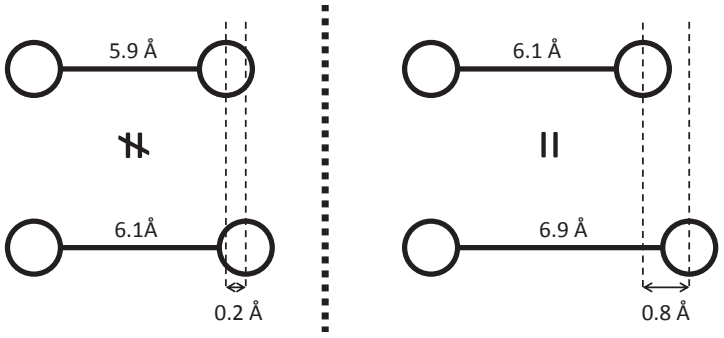


Figure 5.1: Example for the discontinuity on bin-boundaries problem: In the left case the difference between both distances is 0.2 Å, however, due to the use of bins $]a, a+1], a \in \mathbb{N}$, both distances are considered unequal, whereas in the right case the distances are considered as a match even though their difference is much larger (0.8 Å).

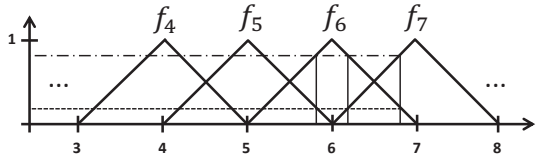
produce a significant change in the shape of the histogram. These problems are referred to as the *discontinuity on bin-boundaries* problem which is illustrated in Figure 5.1. In image retrieval, this problem was reported first (Siggelkow and Burkhardt, 2002), moreover it was shown that it cannot be solved completely

by defining specialized distance measures on histograms (cf. Section 5.3). Instead, the problem must be considered already at the level of the construction of histograms (Vertan and Boujemaa, 2000). Therefore, Vertan and Boujemaa (2000) proposed using techniques from the field of fuzzy logic to overcome the discontinuity on bin-boundaries problem. Fuzzy histograms are intended to be more robust in this regard, especially in the presence of noisy data. For the application considered here, this is especially important, since distances between pseudocenters can vary due to measurement errors or biological variability. Moreover, fuzzy histograms have a smooth instead of a discontinuous shape, which is often more convenient.

The basic idea of fuzzy histograms is to replace bins by fuzzy bins b characterized by fuzzy subsets $X[b]$ of $\mathcal{X}$. A fuzzy partition of a domain $\mathcal{X}$ is defined by a finite family of fuzzy subsets $X[1], X[2], \ldots, X[k]$ of $\mathcal{X}$, such that $\sum_{i=1}^{k} X[i](x) > 0$ for all $x \in \mathcal{X}$; typically, one even requires that $\sum_{i=1}^{k} X[i](x) = 1$ for all $x \in X$. In the concrete case of multisets $\mathcal{D}$ or $\mathcal{D}_{i,j}$, representing distances, a generalized fuzzy partition is used in which $X_\sigma[i]$ is defined by

$$X_\sigma[i](d) = \begin{cases} \frac{1}{\sigma} \cdot d + 1 - \frac{i}{\sigma} & \text{if } i - \sigma \leq d \leq i \\ -\frac{1}{\sigma} \cdot d + 1 + \frac{i}{\sigma} & \text{if } i < d \leq i + \sigma \\ 0 & \text{otherwise} \end{cases}, \tag{5.1}$$

for a certain realization of the parameter σ giving the width of the fuzzy sets. Thus, $X[b]$ can be interpreted as the fuzzy subset of numbers "approximately equal to b" and each element $d \in \mathcal{D}$ belongs to a bin b to the degree $X[b](d) \in [0,1]$. The fuzzy histogram itself, h_f, is then defined as a $\mathcal{B} \longrightarrow \mathbb{R}$ mapping in



	5	6	7
d = 5.9	0.1 (0)	0.9 (1)	0.0 (0)
d = 6.1	0.0 (0)	0.9 (0)	0.1 (1)
d = 6.1	0.0 (0)	0.9 (0)	0.1 (1)
d = 6.9	0.0 (0)	0.1 (0)	0.9 (1)

Figure 5.2: Construction of the fuzzy feature vectors (left) and resulting vectors (right) for different observed distances d, where numbers given in brackets are entries in the resulting crisp feature vector with bins $]a, a+1], a \in \mathbb{N}$.

a straightforward way, namely by replacing counts with sigma-counts. Thus,

$$h_f(b) = \sum_{d \in \mathcal{D}} X[b](d) \ ,$$

with $\mathcal{D}$ the given set of data, hence observed distances.

The advantage of this procedure is illustrated in Figure 5.2, where two comparisons are performed, namely the comparison between the length 5.9 and 6.1 and another comparison between 6.1 and 6.9. The fuzzy histograms of 5.9 and 6.1 share considerably more similarity than the fuzzy histograms representing 6.1 and 6.9. In case of crisp histograms, a different picture can be observed, where histograms representing 5.9 and 6.1 are different, and those representing 6.1 and 6.9 are equal.

Complexity

The time needed for the construction of fuzzy histograms approximating the distributions of the multisets $\mathcal{D}$, respectively $\mathcal{D}_{i,j}$, depends on the number of pseudocenters n and the width of the fuzzy membership functions which is responsible for the number l of overlapping fuzzy sets. However, the widths used in this thesis will be very small and can be neglected. E.g., in the extreme case, where $\sigma = 1$, for each element there are exactly two fuzzy sets supporting it, hence the parameter l becomes a constant. As a result, a time complexity of $\mathcal{O}(n^2)$ is again obtained. Since the histograms created thus are still of size $d_{\max}$, the space complexity also remains unchanged.

5.2 Feature Vectors

The feature vector approach is quite similar to the histogram approach, more precisely, histograms can be seen as a special case of feature vectors. The main difference is that more complex patterns are considered which are moreover not scaled metrically. The concept of feature vector allows one to define a set of features that are specialized for a certain domain. Since there are no restrictions for the considered features, a very large number of such features can in fact be considered. As already mentioned at the beginning of this chapter, one has to find a trade-off between effectiveness and efficiency. Quite simple features were already considered in Section 5.1, since the observed physicochemical properties and distances can be treated as features. As already mentioned,

these features, however, might not be the best choice to represent geometric data. Therefore, 3-simplices are considered here as very informative features for modeling protein binding sites.

5.2.1 Simplices

One-dimensional features for three-dimensional objects might not be the best choice, since they are not able to represent the surface of these objects; this is why 3-simplices are considered here. A set of three points that are linearly independent is called 3-simplex. Such points obviously span an area, thus are very interesting features for the comparison of protein binding sites because they are able to represent certain parts of the surface of the binding site, i.e., parts generated by triangularization of a curved surface. As a drawback and in contrast to the features introduced first, the number of features, therefore the runtime becomes high, as will be shown later.

Defining 3-simplices as three points in the Euclidean space and using them as features comes with many problems, e.g., because protein binding sites do not have a common coordinate system. As an alternative, distances between these points can be used as coordinates to describe 3-simplices resulting in a rotation- and translation-invariant coordinate system and, according to the definition used here, to graphs of size three. Therefore, all non-isomorphic graphs of size three could be considered here as features, where the edges are not labeled with elements of $\mathbb{R}$, but instead where intervals are assigned of the form $]i, i+1]$, for $i = 0, \ldots, k$. The reason for discretization is again the need to have a small set of patterns and the fact that the probability to observe a certain distance from $\mathbb{R}$ is zero.

In the general case, assuming $n_{\mathcal{L}}$ distinct node and k distinct edge labels (given in terms of intervals as defined above), there exist

$$N(n_{\mathcal{L}}, k) = \binom{n_{\mathcal{L}}}{3} \cdot k^3 + n_{\mathcal{L}} \cdot (n_{\mathcal{L}} - 1) \cdot k \cdot \binom{k+1}{2} + n_{\mathcal{L}} \cdot \binom{k+2}{3}$$

features of this type, which can be verified by means of a case distinction:

1. All three node labels are distinct: There are $\binom{n_{\mathcal{L}}}{3}$ possibilities to choose 3 distinct labels from a set of $n_{\mathcal{L}}$ labels. Moreover, since edges are ordered uniquely in this case, there are k^3 possibilities for the edge labels.

2. Two node labels are equal and different from the third: There exist $n_{\mathcal{L}} \cdot (n_{\mathcal{L}} - 1)$ possibilities to choose the two labels, one for the identically la-

beled nodes and one for the other. Assuming an arbitrary ordering on the nodes and edges, an isomorphism can switch the equally labeled nodes so that the ordering of two edges will also change. To map isomorphic graphs uniquely, edges are sorted, which leads to $k \cdot \binom{k+1}{2}$ possible edge combinations.

3. All nodes have identical labels: An isomorphism can reorder all nodes in this case. Therefore, to obtain a unique representation of the possible graphs, all edges must be sorted according to their label. Thus, there are $n_{\mathcal{L}}$ possible node labels and $\binom{k+2}{3}$ edge combinations.

However, considering the set of features as defined above comes with some problems. First of all, not all generated features are simplices, a problem that is of course more of technical nature. A more important problem is the fact that most of the features as defined above are not objects from the 3-dimensional Euclidean space, hence are not observable in the data. E.g., a graph of size 3 exhibiting the sides $(1, 1, 5)$ cannot represent a triangle in the 3-dimensional space. Since pseudocenters are embedded in this space, even the transformation into graphs (using the Euclidean distance to calculate edge weights) does not allow for constructing such graphs of size 3. Hence, many of the patterns thus generated would become unobservable and could skew the obtained measure. Moreover the whole approach would become inefficient in both time and space. Therefore, the set of features is reduced to such patterns that are observable in the data, hence to objects in the 3-dimensional Euclidean space.

Observable Patterns

A triangle with fixed sides a and b gives one degree of freedom, namely the angle α between a and b, as illustrated in Figure 5.3. This property allows one

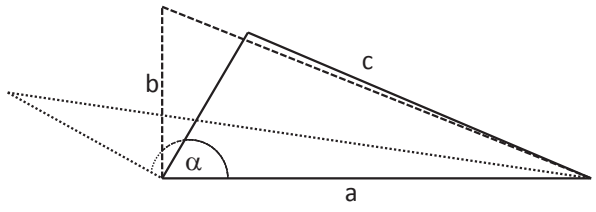


Figure 5.3: By fixing the lengths a and b, the angle $\alpha \in]0, \pi[$ enables one degree of freedom that allows variation of the length c.

to derive conditions that must be fulfilled to ensure that the pattern generated is observable. Fixing e.g. the sides a and b allows a certain length for the side c. By setting $\alpha = 0$ the side c must have length $a - b$, which, however, can be increased up to $a + b$ by choosing larger angles α (in its other extreme $\alpha = \pi$). Following this observation allows the definition of 3 rules, each of which must be fulfilled to ensure that the considered combination of 3 edge intervals $]a, a+1]$, $]b, b+1]$ and $]c, c+1]$ contains at least one triangle that is observable. These rules are given by:

$$\begin{aligned} (a+1)+(b+1) &\geq c\,, \\ (a+1)+(c+1) &\geq b\,, \\ (b+1)+(c+1) &\geq a\,. \end{aligned} \tag{5.2}$$

The original set of patterns having cardinality $N(n_{\mathcal{L}}, k)$ is reduced strongly by applying the rules in (5.2) leading to a much smaller set of patterns, each of which is theoretically observable.

To estimate this number of observable patterns $N_o(n_{\mathcal{L}}, k)$, first the number $n_{\mathcal{L}}$ of distinct node labels is taken out of consideration and only the number k of distinct edge labels is considered. By applying the rules in (5.2) for fixed side intervals $]a, a+1]$ and $]b, b+1]$ a certain maximum length for the remaining side is obtained, which is given by $a + b - 1$ that however has to be bounded by $k - 1$, since $]k-1, k]$ is the maximal side length overall allowed. Hence, the third side length can be in the interval $[c, c+1]$, where

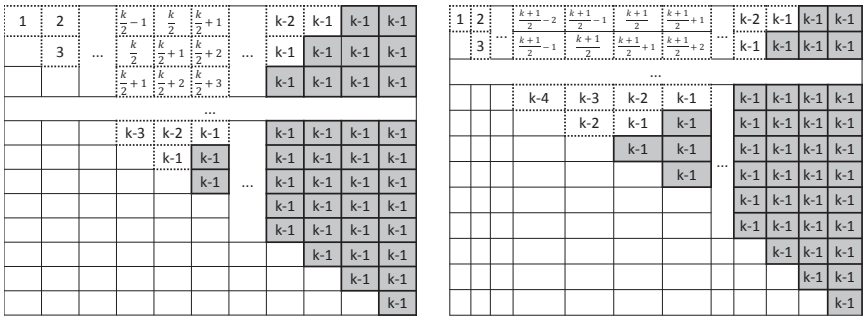
$$c = \min\{a + b + 1, k - 1\}\,.$$

By taking into consideration isomorphism that is expressed in the form of $a \leq b \leq c$, the overall number of observable triangles is given by

$$\begin{aligned} & \sum_{a=0}^{k} \sum_{b=a}^{k} \sum_{c=b}^{\min\{a+b+1,k-1\}} 1 \\ = \;& -\frac{1}{3}k^3 - \frac{1}{2}k^2 + \frac{5}{6}k + 1 + \sum_{a=0}^{k} \sum_{b=a}^{k} \min\{a+b+2, k-1\} \\ = \;& -\frac{1}{3}k^3 - \frac{1}{2}k^2 + \frac{5}{6}k + 1 \\ & + \begin{cases} \sum_{i=1}^{\frac{k-1}{2}} \sum_{j=2i}^{k} (j-1) + \sum_{i=0}^{\frac{k-1}{2}} (2i+1)(k-1) + (k+1)(k-1) & \text{k is odd} \\ \sum_{i=1}^{\frac{k}{2}} \sum_{j=2i}^{k} (j-1) + \sum_{i=1}^{\frac{k}{2}} (2i)(k-1) + (k+1)(k-1) & \text{k is even} \end{cases} \end{aligned}$$

$$= \begin{cases} \frac{1}{12}k^3 + \frac{5}{8}k^2 + \frac{5}{12}k - \frac{1}{8} & \text{k is odd} \\ \frac{1}{12}k^3 + \frac{5}{8}k^2 + \frac{5}{12}k & \text{k is even} \end{cases}. \tag{5.3}$$

In (5.3) in the double-sum $\sum_{a=0}^{k} \sum_{b=a}^{k} \min\{a+b+2, k-1\}$ a minimum must be evaluated, which makes a transformation into a closed expression very difficult. Therefore, a case-distinction is used, where the sum is considered respectively for odd and even k. Moreover, the structure of the sum is analyzed, which is depicted in Figure 5.4. Considering this structure, the minimum operator can be substituted by two sums, each of which counts the cases in which



(a) structure for even k (b) structure for odd k

Figure 5.4: The sum $\sum_{a=0}^{k} \sum_{b=a}^{k} \min\{a+b+2, k-1\}$ represented in the form of a matrix: The variable a is varied in the vertical and b in the horizontal direction. In the case that the minimum is taken in the first argument, the cell is framed with a dotted line, otherwise, bold lines and a gray cell color is used.

respectively the first argument or the second argument is returned by the minimum.

The number obtained by (5.3) is very small, however, so far node labels have not been taken into consideration. This is finally done by considering all ordered combinations of 3 labels from the set of labels $\mathcal{L}_V$ whose cardinality is given by $n_{\mathcal{L}}$. Obviously, this number of combinations is given by $n_{\mathcal{L}}^3$. Thus, the number of patterns considered in this approach is

$$N_o(n_{\mathcal{L}}, k) = \begin{cases} \left(\frac{1}{12}k^3 + \frac{5}{8}k^2 + \frac{5}{12}k - \frac{1}{8}\right) \cdot n_{\mathcal{L}}^3 & k \text{ is odd} \\ \left(\frac{1}{12}k^3 + \frac{5}{8}k^2 + \frac{5}{12}k\right) \cdot n_{\mathcal{L}}^3 & k \text{ is even} \end{cases}.$$

Given 11 distinct edge labels and 7 node labels as used in the following of this thesis, the number of features could be reduced by 17.15% in comparison to the naive approach in which all triangles are considered.

Deriving Feature Vectors

To derive a feature vector for a protein binding site, it is transformed to a complete node-labeled and edge-weighted graph, as described in Section 2.1.2. Since the defined features contain a discrete set of edge weights, the graph representation must be adapted, namely by discretizing edge weights to the set of intervals already used for the construction of the 3-simplices, namely intervals $]a, a+1]$. After transforming the protein binding site into a graph and after discretizing the edge weights, a large number of subgraph-isomorphism problems is solved. The feature vector of a graph G therefore is defined as

$$f_G = \left(G \sqsupseteq t_1, G \sqsupseteq t_2, \ldots, G \sqsupseteq t_{N_o(n_{\mathcal{L}},k)}\right) \in \mathbb{N}^{N_o(n_{\mathcal{L}},k)} ,$$

where $\{t_1, \ldots, t_{N_o(n_{\mathcal{L}},k)}\}$ is the set of all observable non-isomorphic subgraphs of size three, numbered in an arbitrary but fixed order since no natural order can be applied. The predicate $G \sqsupseteq t_i$ tests whether t_i is contained in G and returns the number of occurrences.

The number $N_o(n_{\mathcal{L}}, k)$, even after the unobservable graphs of size 3 are removed, can obviously become very high, the test for subgraph isomorphism moreover is expensive. Therefore, at first sight, this procedure seems to be very inefficient. To accelerate the calculations, a smart indexing function is defined that reduces the runtime dramatically.

Index Function and Resulting Complexity

Using a brute-force implementation to determine the feature vector would become very inefficient. However, the shape of the patterns considered here can be exploited to accelerate the calculation of the feature vectors. The patterns used, graphs of size three, can be described in a canonical form. To this end, three cases are distinguished:

1. All node labels are equal. In this case, the canonical form is given by the node label followed by the edge labels, hence identifiers k of the intervals $]k, k+1]$ in increasing order.

2. Two nodes have an identical label. The canonical form starts with the node label that appears once in the graph followed by the label that appears twice, the edge label between the nodes with the same label, and finally the remaining two edge labels in increasing order.

3. All nodes have different labels. The canonical form is then defined by the three occurring labels, sorted in a lexicographic order, the edge label between the first and the second, the second and the third, and finally the first and the third node.

All three cases are illustrated by an example in Figure 5.5. The set of all canon-

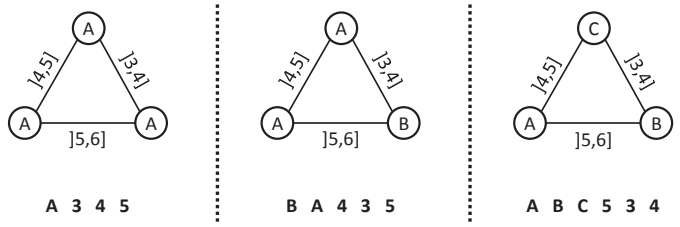


Figure 5.5: The three possible cases that can occur: Graph of size 3 having all labels identical; a graph of same size with two identical labels; finally a graph having three unique labels.

ical forms is denoted by Σ, on which a bijective function is defined, namely $i : \Sigma \to \{1, \ldots, N_o(n,k)\} \subset \mathbb{N}$, which assigns a unique number to each form and therefore subgraph of size three.

The function i allows the substitution of the brute-force approach in which the occurrence of each pattern in a graph is counted separately by solving subgraph isomorphism problems. Using i, the calculation of the feature vector for a graph $G = (V, E)$ can be done in a more efficient way, as all subgraphs of size three in G can be enumerated.

Complexity

For each subgraph g_i of G, the transformation to its canonical form σ_i is performed in time $\mathcal{O}(1)$, the function $i(\sigma_i)$ is evaluated to determine the position of g_i in the feature vector again in time $\mathcal{O}(1)$, and finally the entry at this position in the vector is incremented. Doing this for all $\binom{|V|}{3} = \mathcal{O}(|V|^3)$ subgraphs of size 3 leads to a runtime complexity of $\mathcal{O}(|V|^3)$. Beside this gratifying time complexity an acceptable space complexity of $\mathcal{O}(N_o(n,k))$ is obtained, since vectors of size $N_o(n,k)$ are considered.

5.2.2 Fuzzy Simplices

The features considered in the crisp case contain distances between pseudo-centers that were discretized by assigning them to a certain bin. As already in

the case of distance multisets $\mathcal{D}$ for histograms, this discretization comes with the problem of discontinuity on bin boundaries (cf. Figure 5.1). To avoid this problem, the same technique can be used as for histograms. Again, the basic idea is to replace bins by fuzzy bins b characterized by fuzzy subsets $X[b]$ over the domain $\mathcal{X} \subset \mathbb{N}$, as defined in (5.1). The feature vectors eventually obtained are fuzzy in the sense of having entries in the unit interval $[0,1]$. This is intuitively a more natural model and, from a machine learning point of view, may increase the discriminatory power of the feature vectors.

A pattern t is now a graph $G_t = (V_t, E_t)$ of size 3 whose nodes are labeled as before, but whose edges are labeled with fuzzy numbers of the form $X[b]$. A subgraph $S = (V_S, E_S)$ of a graph G with real-valued edge lengths can be isomorphic to a pattern t to a certain degree. For a node $v_i \in V_S$, let $a_i = \ell_V(v_i)$ be the label of the i-th node. Accordingly, for nodes $v_i, v_j \in V_S$, let $x_{i,j} = \ell_E(v_i, v_j)$ be the weight of the edge between these nodes. Likewise, for a node $v_i \in V_t$, let $b_i = \ell_V(v_i)$ be the label of that node, and $F_{i,j}$ the fuzzy set describing the length of the edge between node i and node j in the pattern t. To determine the degree of isomorphism of t and S, all permutations of the nodes in S are considered, each of which defines a one-to-one correspondence between the nodes of t and S. For each permutation a test is performed to decide if the nodes mapped onto each other share equal label. If the test is successful, the degree of "equivalence" of this permutation depends on the edges mapped (indirectly) onto each other; otherwise no further calculations are necessary and an equivalence of zero is obtained for the permutation considered. Suppose now, the consideration of edges becomes necessary and an edge $x \in E_S$ is mapped onto another edge $F \in E_t$. To evaluate the degree of equivalence of this mapped edges the membership degree $F(x)$ is calculated. Since in sum three edge mappings must be considered, the final equivalence degree is obtained by taking the minimum over the three values, which realizes a generalized conjunction. The degree of isomorphism, denoted $[t \sim S]$, is finally obtained by taking the maximum over all permutations. Hence, this degree is formally defined by

$$\max_{\pi \in S_3} \begin{cases} \min\{F_{12}(y_{12}), F_{13}(y_{13}), F_{23}(y_{23})\} & \text{if } M(\pi) \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

where $y_{i,j} = x_{\pi(i),\pi(j)}$, S_3 is the set of all permutation $\{1,2,3\} \to \{1,2,3\}$, and

$M(\pi)$ is true if

$$(a_1 = b_{\pi(1)}) \wedge (a_2 = b_{\pi(2)}) \wedge (a_3 = b_{\pi(3)})$$

and false otherwise. Likewise, the degree to which t is present in the graph G is then given by

$$[G \sqsupseteq t] = \max_S [t \sim S] \ , \tag{5.5}$$

where the maximum is taken over all subgraphs of size 3 in G. This value defines the entry for the pattern t in the (fuzzy) feature vector $[f_G]$ of G. Hence, the construction of fuzzy feature vectors is analogous to the non-fuzzy approach, with the exception that corresponding entries in the vector are not simply incremented for each subgraph g_i but instead updated according to (5.5).

Complexity

For fuzzy feature vectors, the complexity increases in comparison to the crisp case. Even though (5.4) can be evaluated in constant time, the calculation of (5.5) takes $\mathcal{O}(|V|^3)$ for each position in the feature vector, because there are $\binom{|V|}{3}$ subgraphs of size three that must be considered. In sum, since vectors of size $\mathcal{O}(N_o(n_\mathcal{L}, k))$ are considered, the time complexity becomes $\mathcal{O}(N_o(n_\mathcal{L}, k) \cdot |V|^3)$, where of course N_o can be considered as constant according to the $\mathcal{O}$ notation. The space complexity remains $N_o(n_\mathcal{L}, k)$.

5.3 Measures

So far, it was shown how protein binding sites can be transformed into histograms or vectors, to reduce the problem of comparing complex objects to a comparison of mathematical structures for which a measure can be calculated in a simpler way. After application of the techniques presented above, measures on histograms or vectors can be applied to compare protein binding sites. In the literature, a vast number of methods are given to compare (feature) vectors and histograms. In this thesis, an exhaustive overview cannot be given, instead, one is referred to Deza and Deza (2009) who give a comprehensive introduction. Here, important measures will be recalled that are especially appropriate for the comparison of histograms or feature vectors derived from protein binding sites.

5.3.1 Measures for Histograms

Having reduced the representation of a protein binding site to a set of histograms, the problem of comparing different binding sites can be solved by defining proper measures on histograms. More specifically, consider two binding sites with histogram representations (g_1, g_2) and (h_1, h_2), respectively, thus with the representation in which the physicochemical properties and distances are considered separately. Moreover, suppose that δ_1 is a distance measure suitable for comparing physicochemical property histograms g_1 and h_1, and that δ_2 is a distance measure suitable for comparing distance histograms g_2 and h_2. The overall distance between the two binding sites can then be defined, for example, by

$$\sqrt{\delta_1(g1, h_1)^2 + \delta_2(g_2, h_2)^2}\,,$$

i.e., by the L_2-norm of the tuple of distances. Similarly, for the pairs representation, a measure of the form

$$\sqrt{\sum_{i=1}^{28} \delta(g_i, h_i)^2}$$

can be used. Here, only a single distance δ is needed, since all histograms are of the same type.

In the literature, two types of distance measures are distinguished, namely *bin-by-bin* and *cross-bin* measures. The former are rather simple and only compare the values in the same bin. The overall distance between two histograms is then defined by the sum of distances for all bins. Cross-bin measures, on the other hand, also compare values in different bins. In order to aggregate these distances, these measures also require the existence of a *ground distance* on the set of bins $\mathcal{B}$.

Bin-by-Bin Measures

In the following, two important bin-by-bin measures suitable for comparing two histograms g and h are recalled, both defined on the same set of bins $\mathcal{B}$. Since both measures perform a bin-wise comparison using elementary arithmetic operators, these measures do not influence the time- and space complexity of the overall approach. Hence, the overall complexity for the comparison of two protein binding sites can be taken directly from the method used to generate the histograms.

- *Minkowski Distance:* The well-known Minkowski distance (L_p-norm) is defined as

$$d_M(g,h) = \left(\sum_{b \in \mathcal{B}} |g(b) - h(b)|^p \right)^{\frac{1}{p}}$$

and requires the specification of the parameter p, which is often specified as $p = 1$, $p = 2$ or $p = \infty$. These values will be tried in the experimental study, for the problem of measuring the distance between protein binding sites [1].

- *Histogram Intersection:* The histogram intersection is a measure that comes with the main advantage of an ability to handle partial matches, i.e. when the sum of bins in the histograms considered differs strongly. This problem does not occur in the histograms considered here, since normalized histograms are used that always have an equal sum of bins, namely 1. However, the histogram intersection, in the general case also called the Jaccard coefficient, comes with the advantage of a normalized measure, thus a measure that generates similarities in the interval $[0,1]$ whose interpretation often is more convenient.

  The Jaccard coefficient (in bioinformatics also called Tanimoto index) is defined as a similarity measure between two sets A and B in the form of the fraction

$$\mathrm{sim}(\mathrm{A},\mathrm{B}) = \frac{|A \cap B|}{|A \cup B|} .$$

  For histograms, it must be adapted by substituting the operators cardinality, union and intersection by analogous operators applicable on histograms. Here, cardinality can be expressed as a sum over all bins and union or intersection by the maximum or minimum, respectively. To ensure a uniform class of measures, the Jaccard coefficient is transformed into a distance by defining

$$d_J(g,h) = 1 - \frac{\sum_{b \in \mathcal{B}} \min(g(b), h(b))}{\sum_{b \in \mathcal{B}} \max(g(b), h(b))} .$$

  Again, this measure has complexity linear in the number of bins and does not influence the complexity of the overall approach.

  Apart from these two, many other measures could of course be used as well, for example the χ^2 statistic, the Kullback-Leibler-divergence, etc.

[1] For $p = \infty$ the Minkowski distance becomes $d_M(g,h) = \max_{b \in \mathcal{B}} |g(b) - h(b)|$.

However, it seems that these measures are not appropriate, mainly since they cause some computational problems. In particular, their computation becomes numerically unstable for small values $h(b)$ close to 0. In the application considered here, the probability of encountering such values, or even bins that are completely empty, is rather high.

Cross-Bin Measures

Bin-by-bin measures essentially treat a histogram as a *set* of unrelated bins. Obviously, this comes with a loss of information if, as in the case of distances between pseudocenters, the underlying domain $\mathcal{D}$, on which the bins are defined, is endowed with a metric structure. In this case, it makes sense to consider two bins as neighbored, or to say that bin a is closer to bin b than to c. Cross-bin measures are able to take such relationships between bins into account, which is especially advantageous in the presence of noisy data, where an observation may miss its true bin and instead fall into a neighbored bin. The costs to pay (at least in some cases) are a higher time and space complexity of the overall approach. In the following, some measures of this kind are presented together with their complexity.

- *Quadratic Form Distance:* Given an order $b_1 < b_2 < \ldots < b_n$ on $\mathcal{B}$, a histogram can be written as a vector

$$\boldsymbol{h} = (h(b_1), h(b_2), \ldots, h(b_n))^T \ .$$

Using this representation, the quadratic form distance is defined as

$$d_{QF}(g,h) = \sqrt{(\boldsymbol{g}-\boldsymbol{h})^T A (\boldsymbol{g}-\boldsymbol{h})} \ , \tag{5.6}$$

where A is a matrix whose entries $a_{i,j}$ specify the similarity between bins b_i and b_j. Defining the distance $d_{i,j}$ between bin b_i and b_j by the distance between the corresponding cores (mid-points of intervals in the non-fuzzy case), i.e., $d_{i,j} = |i-j|$, the matrix A can be defined according to (Rubner et al., 2000) by $a_{i,j} = 1 - \frac{d_{i,j}}{\max_{k,l}\{d_{k,l}\}}$. As can be seen, (5.6) performs an all-vs-all comparison of bins, weighting the comparison between b_i and b_j by $a_{i,j}$.

The quadratic form distance requires the specification of a matrix which has in this case a size of $(d_{\max} \times d_{\max})$, on which two vector / matrix multiplications are performed. Multiplication of an $n \times n$ matrix with a vector of size n has complexity $\mathcal{O}(n^2)$. However, since $d_{\max}$ is usually much

smaller than the number of pseudocenters in a protein binding site, the overall complexity is not influenced, as it is still dominated by the algorithm constructing the histograms. The space complexity of $\mathcal{O}(d_{\max}^2)$ is usually very small and can be neglected.

- *Cumulative Distributions:* Another possibility of exploiting an order on $\mathcal{B}$ is to replace the original histogram h by the corresponding cumulative distribution, defined by

$$H(b) = \sum_{a \leq b} h(a) \ ,$$

and to measure the distance on these cumulative distributions. Specifically, the L_1-norm

$$d_M(g,h) = \sum_{b \in \mathcal{B}} |G(b) - H(b)| \ ,$$

is called the *match distance* (Werman et al., 1985), and the L_∞-norm

$$d_{KS}(g,h) = \max_{b \in \mathcal{B}} \{|G(b) - H(b)|\}$$

the *Kolmogorov-Smirnov Distance*. Obviously, as in the case of bin-by-bin measures, the complexity for construction of histograms dominates the whole approach, hence the overall complexity is not influenced by the two measures.

- *Earth Mover's Distance:* The so-called "Earth Mover's Distance" (EMD) is based on the metaphor of moving masses (of earth) from one bin to another one, measuring the corresponding amount of work in terms of the product of mass and distance. The distance between two histograms is then defined by the minimum amount of work needed to transform the first histogram into the second.

  It is not difficult to see that the problem of computing such a distance can be formalized as a min-flow problem (answering the question which part of the mass $g(b_i)$ should be moved to $h(b_j)$ and vice versa) and, therefore, takes the form of a quadratic program (QP).

  The original problem formulation has a rather high memory requirement, due to the need to store a large number of constraints, which is problematic here, because a very large set of constrains is obtained. Fortunately, (Okada and Ling, 2007) proposed an efficient algorithm that makes even

the problem considered here amenable to the EMD. Using the L_1-norm as ground distance on $\mathcal{B}$, the problem of calculating the EMD still remains a QP, however, with a formulation that is much more compact:

$$d_{EMD}(g,h) = \begin{cases} \min\left\{\sum_{\mathcal{B}_n} f_{i,k} \,\middle|\, \{f_{i,k} : (i,k) \in \mathcal{B}_n\}\right\} & \\ \text{subject to:} & \\ \sum_{k:(i,k)\in\mathcal{B}_n}(f_{i,k} - f_{k,i}) = g(b) - h(b) & \forall\, b \in \mathcal{B} \\ f_{i,k} \geq 0 & \forall\, (i,k) \in \mathcal{B}_n \end{cases} \tag{5.7}$$

The corresponding program, given in (5.7), can again be solved with standard QP-solvers. The idea behind the simplification is that it suffices to consider the so-called neighbor-flows between adjacent bins, since all other flows can be replaced by a cost-equivalent sequence of neighbor-flows. Therefore, the QP only considers flows $f_{i,k}$ with $|i-k| = 1$.

The earth-movers distance seems a very interesting approach to measure distance between data that are subject to noise and mutations. The measure however comes with an exponential runtime, since quadratic programming is NP-hard in the general case (Nocedal and Wright, 2000).

5.3.2 Measures for Feature Vectors

For feature vectors, at least for those features considered here, it makes no sense to consider cross-bin measures, since a ground distance cannot be derived from the features. Instead, these vectors are often used together with kernel functions, a combination which is often referred to as a fingerprint kernel. Although the goal of this thesis is not to define kernel-functions, kernels on vectors are used as a class of widely-accepted similarity measures in this domain. Two promising candidates are the Hamming similarity and the Jaccard coefficient, whose advantages have already been described when introduced as a measure for histograms. Again, due to the performance of bin-wise comparisons, the time and space complexity is dominated by the feature vector creation.

- Hamming Similarity: The simplest approach to compare vectors is to

look for the Hamming similarity of the two vectors, which leads to

$$k_{FPH}(G, G') = \frac{1}{N_o(n,k)} \sum_{i=1}^{N_o(n,k)} \kappa_\delta([f_G]_i, [f_{G'}]_i) \ , \tag{5.8}$$

where $[f_G]_i$ denotes the i-th entry in the vector f_G, and κ_δ is the Dirac kernel, i.e.

$$\kappa_\delta(x, y) = \begin{cases} 1 & x = y \\ 0 & \text{otherwise} \end{cases} .$$

Yet a potential disadvantage of this approach is that it does not only reward the co-occurrence of a substructure in both graphs, but also the simultaneous absence: If the i-th pattern neither occurs in G nor in G', then $\kappa_\delta([f_G]_i, [f_{G'}]_i) = \kappa_\delta(0,0) = 1$, which may not be desirable. Moreover, the possible difference in frequency of the respective patterns is neglected.

- Jaccard Coefficient: The Jaccard coefficient is an alternative measure avoiding this problem. It was already used for the comparison of histograms and can also serve as measure on feature vectors without changing its definition. On the feature vectors, the Jaccard coefficient is concretely defined as

$$k_{FPJ}(G, G') = \frac{\sum_{i=1}^{N_o(n,k)} \min([f_G]_i, [f_{G'}]_i)}{\sum_{i=1}^{N_o(n,k)} \max([f_G]_i, [f_{G'}]_i)} \ . \tag{5.9}$$

Measures on Fuzzy Feature Vectors

The values of fuzzy feature vectors range in the interval $[0, 1]$ which makes a comparison based on the Hamming similarity impossible, since it is very unlikely to observe two equal values. A naive approach would be to use bins or to discretize values, however, fuzzy feature vectors were introduced in particular to solve the discontinuity on bin-boundaries problem, which occurs exactly by using one of these two techniques. Hence, such an approach would become counterproductive, one reason why for the comparison of feature vectors only the Jaccard coefficient is considered[2].

In fact the Jaccard coefficient can be directly used as defined in the case of crisp feature vectors with the difference that logical operators are substituted

[2]Even though, there exits further methods which could be applied, e.g. cosine similarity (Deza and Deza, 2009)

by their generalizations. The formal definition therefore becomes

$$k_{FPJ}(G, G') = \frac{\sum_{i=1}^{N_o(n,k)} \top([f_G]_i, [f_{G'}]_i)}{\sum_{i=1}^{N_o(n,k)} \bot([f_G]_i, [f_{G'}]_i)} \ ,$$

where $\top$ and $\bot$ are realizations of a triangular-norm and -conorm, respectively. Thus, (5.9) is a special case in which the minimum t-norm and the maximum t-conorm are used.

5.4 Summary

Many techniques of data analysis, in particular the measurement of similarity or distance between objects, are based on vectors. A well-known example is WEKA (Witten et al., 2011) which is a very powerful *Machine Learning* toolbox that requires data in the form of feature vectors. In this thesis feature vectors are used to simplify the representation of a protein binding site, therefore to enable the definition of measures in an easier way.

The basic idea of deriving feature vectors for a certain complex object is to create a set of patterns, to order them in an arbitrary but fixed way and to count the occurrences of these patterns in the object. In this chapter, different patterns were proposed that can be used to derive feature vectors, beginning with simple one-dimensional patterns that allow a very fast processing of the data, up to 3-simplices that are appropriate to represent the surface of a 3-dimensional object, hence, that should be useful especially for the problem considered here, albeit at the cost of runtime.

Different measures were introduced to compare vectors, hence the objects represented in the form of feature vectors. Moreover, it was shown that for simple patterns which can be observed directly in the data, histograms can be used to approximate the underlying distribution of these patterns. Therefore, for those patterns, histograms are used as a special case of feature vectors with corresponding specialized measures.

An important problem occurring in feature-based approaches was also discussed in this chapter: Since the set of patterns must be finite and should be small, only a small set of edge weights can be covered. Therefore, a discretization must be performed that leads to the problem of discontinuity on bin-boundaries, which is especially problematic in the case of biological data, where mutations can occur and where the data is often subject to noise and

structural flexibility. Methods of fuzzy logic can be used to solve this problem, leading to so-called fuzzy patterns or histograms, used here to enable higher error-tolerance.

The most important advantage a feature-based representation provides is that the feature vector can be calculated once for each protein binding site and used afterwards for many comparisons. This is of course also the case for graphs, however, the construction of graphs is efficient, whereas calculations on graphs are usually very expensive. The calculation of feature vectors in contrast is in some cases of high complexity, the comparison of vectors, on the other hand, is very efficient. Therefore, storing the feature vectors in parallel to pseudocenters would allow the comparison of protein binding sites in the milliseconds range.

6

Graph-based Approaches

Graphs have long been used for many types of problem. Especially in the field of chemoinformatics, molecules are represented in terms of graphs. In bioinformatics, the number of applications in which graphs are used increases continually. This is eventually due to the universal applicability of graphs: They can be used to represent networks, relations and geometric structures. Moreover, graphs are well-studied mathematical structures for which many algorithmic concepts exist. Graph representations have already been used for analyzing protein binding sites, e.g. in (Weskamp et al., 2007; Shulman-Peleg et al., 2004). They are characterized especially by their potential to provide a very flexible model, hence a model that allows for introducing an arbitrary degree of error-tolerance. This high degree of flexibility, however, often comes with a combinatorial character of the resulting algorithms often combined with high complexity.

In this chapter, different classes of methods will be presented: On the one hand approaches that calculate the raw similarity between two graphs, but on the other hand also such methods that calculate more than a similarity value, namely an alignment of the basic elements of the graphs, hence an alignment of nodes and indirectly derived from this alignment, an alignment of edges. Alignments can be further distinguished by their result: Partial alignments align only a subset of the nodes under consideration. Complete alignments, in contrast, set all considered nodes into one-to-one correspondence. In the simplest case, pairwise alignments are considered. To generate multiple alignments, thus alignments of more than two graphs, merging techniques are often applied. Theoretically, these techniques could be also applied on partial alignments, however, the size of the multiple partial alignment would become a

monotonic decreasing function in the number of graphs, thus these alignments would become very small with the increasing number of graphs.

In the following, methods for graph representation are introduced beginning with those approaches that calculate a raw similarity. Parts of this chapter were already published in (Fober et al., 2009b), (Fober et al., 2009c), (Boukhris et al., 2009) and (Fober and Hüllermeier, 2011).

6.1 R-convolution Kernels on Graphs

Kernels are functions $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ that fulfill certain properties, namely symmetry and positive definiteness. These kernels have recently attracted a huge amount of interest since they allow use of the rich repertoire of kernel-based learning techniques (Shawe-Taylor and Cristianini, 2003). The concept of kernel functions was also successfully adapted for graphs, leading to graph kernels. In this thesis, kernel-based learning will not be considered further, a comprehensive introduction on graph kernels can be found in (Gärtner, 2008). Here, kernels will simply be considered as similarity measures without use of their properties.

Generally, it is not simple to define an appropriate similarity measure on graphs, in particular a measure that is error-tolerant and efficient to calculate. However, there exist concepts that allow an easy definition of such a measure. One of these concepts is the *R*-convolution framework (Haussler, 1999) that not only leads to an error-tolerant similarity measures that can be computed in polynomial time, but even guarantees the kernel properties. Graph kernels that are based on the *R*-convolution concept decompose the graphs into a set of simple substructures of a specific type. The comparison of whole graphs is then reduced to the level of these substructures. The idea is that, for such substructures, the definition of adequate similarity measures is less difficult and, hopefully, the computation more efficient.

More concretely, let $\mathcal{G}$ be a set of objects composed of certain substructures, where in this case $G \in \mathcal{G}$ is a graph, and $\tilde{\mathcal{G}}$ a set containing all possible substructures of the graphs in $\mathcal{G}$. Furthermore let $R \subseteq \mathcal{G} \times \tilde{\mathcal{G}}$ be a relation that consists of such tuples (G, g) for which g is a substructure of G. The inverse $R^{-1}(G) = \{g \mid (G, g) \in R\}$ is obviously a valid decomposition of a graph $G \in \mathcal{G}$ into its substructures $g \in \tilde{\mathcal{G}}$. If it is possible to find a function $\kappa : \tilde{\mathcal{G}} \times \tilde{\mathcal{G}} \rightarrow \mathbb{R}$ that fulfills the kernel properties, the *R*-convolution framework

allows definition of a kernel between graphs $G, G' \in \mathcal{G}$ by

$$k(G, G') = \sum_{g \in R^{-1}(G)} \sum_{g' \in R^{-1}(G')} \kappa(g, g') \ . \tag{6.1}$$

Haussler (1999) proved that (6.1) is symmetric and positive definite, thus a valid kernel. Obviously, the *R*-convolution framework is a general framework,

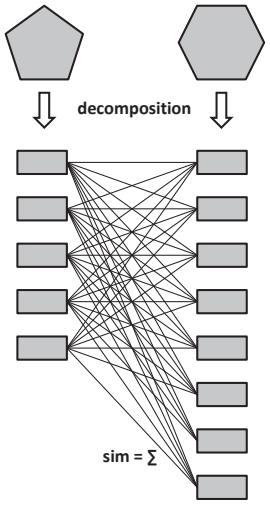


Figure 6.1: Visualization of the R-convolution framework.

not restricted to graphs, and a very useful tool for defining kernels on complex objects. The Figure 6.1 visualizes this procedure: Two objects (pentagon and hexagon), each of which is decomposed into a set of substructures. Elements from these two sets are compared all-vs-all and the distances obtained are summed-up, leading to the final distance.

However, to be able to apply this framework on graphs, an appropriate decomposition technique must be defined. On the one hand, it is of highest importance to ensure that the chosen substructures contain important information, on the other hand, the number of substructures that must be considered should be small so as to guarantee efficiency of the approach. Again, as in the case of feature-based approaches, a trade-off must be found. Two prominent instances of the *R*-convolution framework are the *random walk*- and the *shortest path kernel* (Gärtner, 2003; Borgwardt, 2007). While in the first case the set R^{-1} contains all random walks, it contains in the latter case all shortest paths. Obviously, shortest path kernels come with the advantage that the number of substructures is small, thus the calculation of the kernel becomes more efficient.

As Borgwardt (2007) states, it comes without tottering. Decomposing a graph into its random walks leads to repetition of nodes and edges, thus to an artificially high similarity value. This phenomenon is called *tottering* and is strongly reduced in the case of shortest path kernels. Instead, however, shortest paths may contain less information, e.g., in the case of a complete graph and edge weights based on the Euclidean distance, shortest paths would contain only one edge. Therefore, in advance, there is no conclusion possible if the former or latter approach is a better kernel, thus both kernels will be considered in the following.

6.1.1 Random Walk Kernels

Originally, random walk kernels were introduced in (Gärtner, 2003) for unweighted graphs. As the name suggests, this type of kernel draws random walks to decompose a graph into its substructures. More specifically, this realization of the R-convolution framework decomposes a graph into sequences of nodes generated by these random walks. Instead of drawing all random walks in a graph, which would be computationally infeasible, Gärtner (2003) exploits a property of the product graph $G_\times$, namely that the sum over all entries (i, j) of the n-th power of the adjacency matrix $A_\times$ represents the number of equal walks of length exactly n. Thus, to calculate the number of equal walks of length n in G and G', it is sufficient to sum up $[A_\times^n]_{i,j}$ over all i, j. To calculate equal walks of arbitrary length, one also has to sum-up over all n, thus over $n = 0, \ldots, \infty$. Therefore, elementary matrix operations can be used to evaluate (6.1), namely by

$$k_{RW}(G, G') = \sum_{i=1}^{|V_\times|} \sum_{j=1}^{|V_\times|} \left[\sum_{k=0}^{\infty} \lambda_k \cdot A_\times^k \right]_{i,j} . \tag{6.2}$$

However, the original definition of the categorical product graph does not consider the labeling of a graph, which obviously would lead to a dramatic loss of information in the case considered here, since important information about physicochemical properties and the geometry would not be available during calculation. A simple but effective extension of this approach, however, is to substitute the original definition of the categorical product graph by the extended product graph (Definition 2.16) that takes node labels as well as edge weights into account. Performing this substitution allows one to extend the original random walk kernel in an easy way for node-labeled and edge-

weighted graphs. Similar approaches have already been used, e.g., to classify chemical compounds (Borgwardt and Kriegel, 2005).

The equation (6.2) is a series for which convergence cannot be guaranteed in general. However, it already contains a factor λ_k that can be used as a shrink factor to ensure convergence and to enable furthermore the transformation into a closed expression. For certain choices of λ, the above series thus can be calculated in a simple way. Choosing $\lambda_k = \lambda^k = (1/a)^k$, with $a \geq \max_{v \in V_\times}\{\text{degree}(v)\}$, leads to the geometrical series, and (6.2) reduces to

$$k_{RW_{geo}}(G, G') = \sum_{i=1}^{|V_\times|} \sum_{j=1}^{|V_\times|} \left[(I - \lambda \cdot A_\times)^{-1} \right]_{i,j} . \tag{6.3}$$

Choosing $\lambda_k = \frac{\beta^k}{k!}$ leads to the exponential series and to

$$k_{RW_{exp}}(G, G') = \sum_{i,j=1}^{|V_\times|} \sum_{j=1}^{|V_\times|} \left[e^{\beta \cdot A_\times} \right]_{i,j} .$$

Complexity

As already mentioned in the introduction, decomposing an object into random walks will lead to a high number of substructures, thus the evaluation of (6.1) will become inefficient. In fact, the calculation is not performed in a direct way, i.e. by drawing all random walks. Instead, an indirect way is chosen and elementary matrix operations are performed. However, since the product graph is of quadratic size and matrix inversion as well as matrix diagonalization has cubic complexity, the complexity of the random walk kernel is $\mathcal{O}(n^6)$, with $n = \max\{|V|, |V'|\}$. In practice this complexity is quite high, especially if the goal is to perform a large scale study. Moreover, random walk kernels suffer from a high space complexity of $\mathcal{O}(n^4)$, since calculations are performed on the product graph.

6.1.2 Shortest Path Kernels

The high computational complexity can be attenuated by choosing a decomposition that comes with a smaller number of substructures. Paths and especially shortest paths were found by Borgwardt and Kriegel (2005) to be appropriate substructures. Decomposing a graph into shortest paths does not only reduce the number of substructures, it also suppresses the problem of tottering and

solves the problem of halting, which is the phenomenon whereby larger substructures, such as longer walks in the case of the random walk kernel, are down-weighted by a shrink parameter, although they contain more information.

Originally, the approach of Borgwardt and Kriegel (2005) uses a more general definition of the R-convolution framework, namely

$$k(G, G') = \sum_{g \in R^{-1}(G)} \sum_{g' \in R^{-1}(G')} \prod_{i=1}^{n} \kappa_i(g_i, g_i') \,, \tag{6.4}$$

where the substructures of a graph are divided into different elements (such as nodes and edges) for which individual kernels κ_i are defined. The authors in (Borgwardt and Kriegel, 2005) propose using three kernels κ_i: A (Dirac) kernel on the path-length (in terms of the number of nodes) to accelerate the calculations, since the calculation of the second and third kernel can be aborted if the length does not match. The second and third kernel are realized respectively as a kernel on nodes and edges. On graphs representing protein binding sites, nevertheless, shortest paths will in most cases differ in their size, thus such an approach would sum up zeros. On the other hand, it is difficult to remove the kernel on path-length and instead to define an appropriate and efficient kernel on sequences of node labels or edge weights that differ in size. Therefore, the original realization is again not appropriate for the comparison of node-labeled and edge-weighted graphs representing protein binding sites, a fact that leads to a new realization of the shortest path kernel in this thesis. Here, another representation of shortest paths is used: For two nodes $v_i, v_j \in G$, let $sp(v_i, v_j)$ denote the length of the shortest path (sum of edge weights on the path) between these nodes. A path is thus represented by its length and the labels of the start and the end node while the node labels in-between are ignored. On the node labels, a Dirac kernel

$$\kappa_\delta(x, x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

is used for comparison. For comparing the length of shortest paths, a Gaussian kernel defined as

$$\kappa_G(x, x') = e^{-\frac{\|x - x'\|^2}{2\sigma^2}} \tag{6.6}$$

is applied. Having specified kernels on the substructures, the shortest path

kernel can be defined by again making use of (6.1):

$$k_{SP}(G,G') = \sum_{(v_i,v_j)\in V^2} \sum_{(v'_k,v'_l)\in V'^2} \kappa_\delta(\ell_V(v_i),\ell_V(v'_k)) \cdot \kappa_\delta(\ell_V(v_j),\ell_V(v'_l)) \cdot \kappa_G(sp(v_i,v_j),sp(v'_k,v'_l)) \; . \tag{6.7}$$

In contrast to the original definition of the shortest path kernel, the approach presented here neglects node labels along the path, and instead deals better with paths of different size that maybe appear due to the deletion of a certain pseudocenter caused by mutation. In any case, an additional variant of the shortest path kernel is developed that exploits as much information as possible, hence nodes and edges along the shortest path, and that is furthermore error-tolerant to a maximal degree, i.e., the kernel on shortest path returns values larger than zero even if the length of both substructures does not match. This kernel is based on sequence alignment and is introduced in the following.

Shortest Path Kernel Based on Sequence Alignment

On the one hand, representing a path only by its length and the labels of the start and end node and, correspondingly, using the Dirac (6.5) and the Gauss (6.6) kernels for comparison obviously entails a considerable loss of information. On the other hand, using the whole label information but discarding it if the length of the considered paths does not match does not seem to be the perfect solution either. To investigate whether performance can be improved by taking the labels of intermediate nodes into account, independent of the length of the considered paths, another extension of the shortest path kernel is developed. More specifically, the simple $0/1$ measures (6.5) and (6.6) are replaced by a measure which compares the complete shortest path sequences (SPS). To this end, an SPS $(v_1, v_2, \ldots, v_l)$ is represented in the form of a sequence

$$\big(l_V(v_1), l_E(v_1,v_2), l_V(v_2), \ldots, l_E(v_{n-1},v_n), l_V(v_n)\big)$$

in which node labels and (discretized) edge lengths occur alternately. To compare such SPS, standard methods from sequence analysis can be used. The Needleman-Wunsch algorithm (Needleman and Wunsch, 1970), a well-known approach based on the Levenshtein distance (Levenshtein, 1966) utilizing dynamic programming, has a runtime $\mathcal{O}(l_A \cdot l_B)$, where l_A and l_B is the length of SPS A or B, respectively.

To comply with the requirements of the application considered here, the original dynamic programming approach to sequence alignment is modified as follows: First, recall that the SPS involve two types of symbols, namely node labels and edge weights, where the latter were discretized into bins of size 1 to allow for testing of equivalence. To ensure that the former are not aligned with the latter, which is obviously not reasonable, the cost for an assignment of this type was set to negative infinity. Second, note that long paths including many nodes represent more of the structure of a graph than paths of short length. Therefore, each score (similarity between two shortest paths) is normalized by dividing it by the length of the longest SPS in the graph. This leads to an overweighting of longer sequences since it can be assumed that these carry more meaningful information than short SPS, thus to an elimination of the halting problem. Again, a measure with values between 0 and 1 is obtained, which is used in (6.7) instead of (6.5) and (6.6).

Complexity

The computation of all shortest paths in a graph can be done using the Floyd-Warshall algorithm (Floyd, 1962) in time $\mathcal{O}(n^3)$, where $n = \max\{|V|, |V'|\}$. The procedure subsequently applied depends on the type of calculation. Using the former approach for calculating the shortest path kernel, the shortest-path matrix is used to store the results obtained by the Floyd-Warshall algorithm. In the entry (i, j) this matrix gives the costs of the shortest path from node i to node j. All paths in both shortest path matrices are considered in a pairwise way and compared using κ_δ and κ_G which require time $\mathcal{O}(1)$. Since there are n^4 comparisons to perform, the shortest path kernel requires time $\mathcal{O}(n^4)$.

In terms of runtime, the extension of the shortest path kernel is of course more expensive, since sequence alignment comes with a higher complexity than a simple test for equivalence. As explained above, the similarity between these paths is measured by using dynamic programming (Needleman and Wunsch, 1970), the runtime of which is quadratical in the length of the sequences. Since this length is $\mathcal{O}(n)$, and since there are $\mathcal{O}(n^2)$ sequences in both graphs, the total complexity for the pairwise comparison of all SPS is $\mathcal{O}(n^3 + n^2 \cdot n^2 \cdot n^2) = \mathcal{O}(n^6)$.

As an advantage of the shortest path kernel, calculations are performed directly on the graphs. This allows an efficient storage leading to a space com-

plexity of $\mathcal{O}(n^2)$ given by the size of the adjacency matrix of the larger graph. Even in the case of sequence alignment, the space complexity is not increased, since the maximal length of a shortest path is $\mathcal{O}(n)$. Hence, the size of the matrix needed during calculation of the sequence alignment is again $\mathcal{O}(n^2)$.

6.2 Approximate Maximum Common Subgraphs

It is usually important to explain the calculated similarity values, and particularly to discover patterns that are shared by the graphs compared. With a graph representation of proteins one is interested in discovering common subgraphs or in discovering the largest common subgraph. To find a common subgraph, methods based on clique detection are widely and successfully used. The main idea of these methods is to explore a property of the product graph, namely that the maximum clique in the product graph $G_\times$ of two input graphs G and G' corresponds to the maximum common subgraph of G and G' (Bunke and Shearer, 1998). Such a maximum common subgraph represents a partial alignment, i.e., a one-to-one correspondence of a subset of nodes in the graphs. Detecting an alignment of nodes allows one to derive the alignment of edges in an indirect way. Obviously, such an alignment can also be used to derive the similarity between two graphs: The larger the maximum common subgraph with respect to the size of the larger graph, the more similar both graphs are. In the following, an extension of this approach is presented which solves some problems of measures based on the (non-approximate) maximum common subgraph, in particular the problem of a low error-tolerance.

6.2.1 Relaxation based on Quasi-Cliques

As already mentioned, the maximum common subgraph of two graphs G and G' corresponds to the maximum clique in the product graph $G_\times$ of these graphs. Using the extended product graph allows enrichment of the graph with node labels and edge weights, hence almost all information provided by CavBase can be used during calculations. The complete definition of the product graph is given in Definition 2.16, here, the emphasis lies on the definition of the edge set of the product graph, thus on

$$E_\times = \left\{ ((v_i, v'_j), (v_k, v'_l)) \in V_\times^2 \mid \|\ell_E(v_i, v_k) - \ell_E(v'_j, v'_l)\| \leq \epsilon \right\}, \qquad (6.8)$$

where ϵ is a user-specified threshold. Missing edges in G respectively G' are substituted by edges with weight infinity, where the distance between such edge weights is assumed to be zero, hence such edges are considered as a match.

Having applied a clique-detection algorithm such as the Bron-Kerbosch algorithm (Bron and Kerbosch, 1973) on the product graph, one ends up with a clique $G_C = (V_C, E_C)$ whose set of nodes $V_C \subseteq V \times V'$ is of interest. This set contains pairs $(v_i, v'_j) \in V_C$, where the first element of a pair corresponds to a node of the first graph that is part of the maximum common subgraph, and the second element of the pair accordingly to a node of the second graph which is also member of the maximum common subgraph. Common subgraphs can be aligned, hence the elements $(v_i, v'_j) \in V_C$ give assignments of nodes in G and G' that belong to the optimal partial alignment.

The Definition 2.16 already allows a certain degree of error-tolerance. By introducing the threshold ϵ, a certain degree of structural flexibility can be allowed, since edges are allowed to differ by this threshold at most. Unfortunately, ϵ is a critical parameter: A small ϵ allows the handling of small differences in the structure caused by noise and structural flexibility of proteins. However, it will not allow one to deal with larger structural differences caused by conformational changes. Thus, this approach is tolerant towards possibly numerous though small errors but not towards single though exceptionally large deviations. Choosing a larger ϵ is impracticable since this would result in new problems: Such an approach would become tolerant towards a possibly large number of large errors. Thus even dissimilar graphs would become similar, which is obviously an undesired behavior of a measure, and which would moreover lead to an incorrect alignment. To circumvent this problem therefore the detection of quasi-cliques in the product graph is proposed here, instead of clique detection.

Cliques are the densest form of subgraph, thus complete graphs, since each pair of nodes must be connected by an edge. A quasi-clique is roughly speaking an almost complete graph $G_{QC} = (V_{QC}, E_{QC})$. In the literature, different definitions of quasi-cliques have been proposed. Some of them are based on the degree of the node (Liu and Wong, 2008), calling G a quasi-clique if every node in V is adjacent to at least $\lceil \gamma \cdot (|V| - 1) \rceil$ other nodes. This is the definition adopted in the following. Yet, other definitions do exist, for example referring

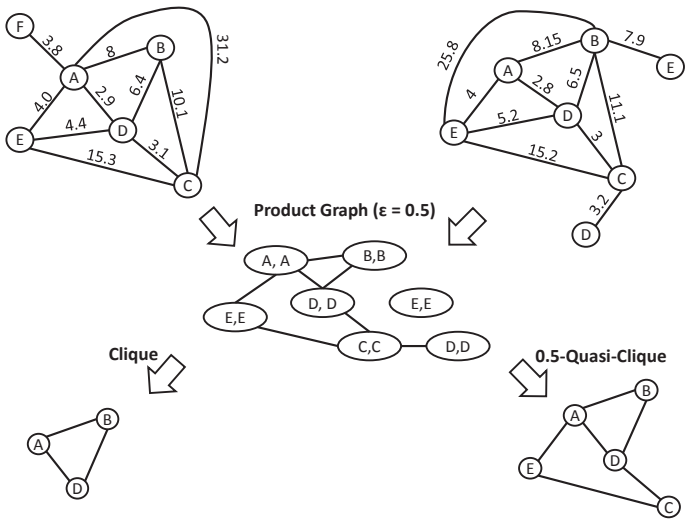


Figure 6.2: Comparison between the clique and quasi-clique approach: The product graph of two input graphs is calculated (middle). This graph, however, is quite sparse, mainly because the nodes labeled with "D" and placed in the middle in both input graphs differ in their position due to a slight shift. The resulting clique hence represents a very small common subgraph. The quasi-clique ($\gamma = 0.5$) instead represents much more structure, especially such a structure one would intuitively identify as the common subgraph of both input graphs.

to edge density (Zeng et al., 2007): A graph is a quasi-clique if $|E| \geq \left\lceil \gamma \cdot \binom{|V|}{2} \right\rceil$. In both cases, $\gamma \in \,]0,1]$ is a relaxation parameter making quasi-cliques to a generalization of cliques, since each clique is a 1-quasi-clique.

6.2.2 Quasi-Clique Detection

As mentioned in Section 2.1.2, the problem of finding a maximum clique in a graph is NP-hard (Karp, 1972). Since a quasi-clique is a generalization of a clique, it immediately follows that finding a maximum γ-quasi-clique is also an NP-hard problem. Therefore, to solve the problem, one has to resort to heuristic algorithms. But the problem of finding γ-quasi-cliques is even more difficult. Heuristic methods for clique detection typically exploit the downward-closure property, namely that a supergraph of a non-clique cannot be a clique either. Unfortunately, this property does not hold for quasi-cliques, as one can easily show by counter-examples, as e.g. in Figure 6.3. Instead, any subset of the set of nodes V in a graph $G = (V, E)$ may form a γ-quasi-clique.

Nevertheless, alternative heuristic methods for quasi-clique detection have

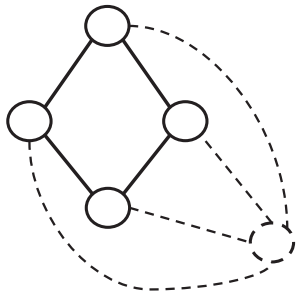


Figure 6.3: Counterexample for the downward-closure property: A partial solution (solid lines) which is not a 0.7-quasi-clique can be extended to such a quasi-clique by including a further node together with its adjacent edges (dashed lines).

been developed. An interesting method was proposed by Liu and Wong (2008), representing all potentially maximal γ-quasi-cliques by its nodes in a set enumeration tree. Thus, the search space is given by the power set of the set of nodes V. Searching for maximal quasi-cliques is performed by means of a depth-first search on the set enumeration tree. Once a quasi-clique has been discovered, it is stored in a prefix tree, so that a maximal γ-quasi-clique is provably found in a leaf of the prefix tree. In this thesis, the technical details are not recalled, instead one is referred to (Liu and Wong, 2008). Here it can be summarized that this approach is very inefficient, and moreover, that the space complexity is also very high. Another way was chosen by (Li et al., 2005) who developed an efficient algorithm that is based on local clique detection for interaction graph mining in protein complexes. This approach is modified technically and it is applied to find approximate common subgraphs, hence it operates on the product graph $G_\times = (V_\times, E_\times)$. Like the approach presented by (Liu and Wong, 2008), this approach allows one to specify the parameter γ giving the minimal density of the clique which is reported as a maximum quasi-clique. However, this approach might miss the correct solution since it is purely heuristic. Basically, this heuristic consists of two steps: The detection of local cliques and a merging procedure.

Local Cliques

To detect a local clique in the product graph $G_\times = (V_\times, E_\times)$, a node $v \in V_\times$ is selected and a neighborhood graph $G_\times^{(v)} = (V_v, E_v)$ is calculated. Until the clique property is not satisfied for $G_\times^{(v)}$, iteratively the node in V_v is removed

that has the smallest degree together with all adjacent edges; in the case that more than one node shares the smallest degree, a node is chosen at random. Once the clique property is fulfilled, a local clique is formed that contains the node v. For testing if a graph G fulfills the clique property, the function dens can be used which returns 1.0 if the graph G is a clique. This function is concretely defined as

$$\text{dens}(G) = \frac{2 \cdot |E|}{|V| \cdot (|V| - 1)} \,. \tag{6.9}$$

Due to the NP-hardness of the problem, cliques found in this step will in most cases neither be the optimal solution for the maximum clique problem nor the maximal clique containing v, as Figure 6.4 illustrates. Here, this approach

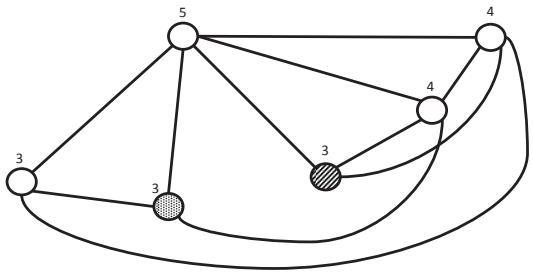


Figure 6.4: Counterexample for the optimality of the local clique approach: The dotted and the lined node both have a degree of 3. If the algorithm chooses at random the lined node, the maximum clique is lost.

is used to generate efficiently a set of cliques that are used in a second step for merging them to quasi-cliques. For this purpose, each node $v \in G_\times$ is considered once and used to create a local clique. These local cliques are stored in a set LC, in which only such local cliques remain that consist of more than 2 nodes.

Local Clique Merging

The merging algorithm requires the specification of three parameters: First of all, γ defines the (smallest allowed) density of the maximum quasi-clique one is looking for. Since the downward-closure property does not hold for quasi-cliques, intermediate solutions with a density below γ cannot be discarded. However, an extension to a γ-quasi-clique becomes unlikely for intermediate solutions whose density is significantly smaller than γ; therefore this condition is tested using a second ("cautious") threshold $\gamma' < \gamma$. Finally, a parameter ω

is used to control the number of merge operations, since the intermediate solutions that are merged must overlap to a degree of at least ω. Having specified these parameters, the cliques in LC are merged iteratively according to Algorithm 6.1 which basically realizes a modification of a beam search (Norvig, 1991) on the search space of local cliques. To this end, a beam C is defined con-

Algorithm 6.1: Merging of Cliques

Input: Set LC of (local) cliques of a product graph $G_\times$

Output: Largest clique of density above γ

Auxiliary function $\mathrm{add}(A, A')$ adds product nodes from A into A', that does not represent (input) nodes which are already contained in A

$\mathrm{largest} = \mathrm{argmax}\{size(G_C) \mid G_C \in LC\}$;

C = LC;

while $C \neq \emptyset$ **do**

  $\mathrm{largest} = \mathrm{argmax}\{size(G_C) \mid G_C \in C \ \wedge \ dens(G) \geq \gamma\}$;

  $C' = \emptyset$;

  for $G_C \in LC$ **do**

    for $G'_C \in C$ **do**

      if $overlap(V_C, V'_C) \geq \omega$ **then**

        $\tilde{V} = \mathrm{add}(V_C, V'_C)$;

        $\tilde{E} = \tilde{V}^2 \cap E_\times$;

        $\tilde{G} = (\tilde{V}, \tilde{E})$;

        if $dens(\tilde{G}) \geq \gamma_1 \ \wedge \ V_C \cap V'_C \neq V_C$ **then**

          $C' = C' \cup \tilde{G}$;

  $C = C'$;

taining the best solutions. The beam is initialized with the set LC containing all local cliques detected in the first step. While the beam is not empty, all pairs (G_Q, G_L) with $G_Q \in C$ and $G_L \in LC$ are considered. If their overlap

$$\frac{|V_Q \cap V_L|}{\min\{|V_Q|, |V_L|\}}$$

is above the threshold ω, the pair is merged and inserted into a temporal set if its density is above γ'. Having considered all pairs, the beam C is replaced by the temporal set and the loop is continued. To avoid the risk of loosing the

best solution[1] found so far, it is stored in a variable which is returned upon termination of the algorithm. The concrete realization of this loop is given in Algorithm 6.1.

Merging is defined as follows: Two graphs G and G' are merged by defining the union of their respective node sets, and connecting a pair in the aggregation $\tilde{V} = V \cup V'$ by an edge, if this edge also exists in the graph $G_\times$; the set $\tilde{E}$ is hence given by $\tilde{V}^2 \cap E_\times$. However, in the case of merging two product graphs, the nodes must be considered more carefully: Since product nodes correspond to unique pairs $(v, v') \in V \times V'$, different product nodes can still correspond to the same nodes in G or G'. This many-to-one relationship must be taken into consideration to ensure that valid approximate common subgraphs are produced. Here, a very simple though efficient procedure is applied, which adds a product node into the quasi-clique only if the nodes it represents are not yet contained.

Generalizing Similarity

Having detected the largest γ-quasi-clique, a similarity measure could be defined by $\mathrm{sim}(G, G') = |V_{QC}| / \max\{|V|, |V'|\}$. However, to reach more flexibility again the framework presented in Section 2.4.1 is used here, to reach a trade-off between two extreme aggregation modes *max* and *min*. The resulting similarity measure is hence defined by

$$\begin{aligned}\mathrm{sim}(G, G') =& \lambda \cdot \min\{|V_{QC}|/|V|, |V_{QC}|/|V'|\} \\ &+ (1-\lambda) \cdot \max\{|V_{QC}|/|V|, |V_{QC}|/|V'|\}\end{aligned}$$

and controlled by the parameter $\lambda \in [0, 1]$.

Complexity

Finding all local cliques in a graphs $G = (V, E)$, where $n = |V|$, requires time $\mathcal{O}(n^3)$. For each of the $\mathcal{O}(n)$ nodes of the graph, the neighborhood graph is constructed in time $\mathcal{O}(1)$ by making use of an adjacency list. The size of the neighborhood graph is $\mathcal{O}(n)$, from which nodes are removed until the clique property holds. To remove a node, first the node with smallest degree v_s is identified in time $\mathcal{O}(n)$ and removed afterwards. Moreover the degree

[1] The best solution is the largest quasi-clique whose density is at least γ.

of all nodes adjacent to v_s is decreased by one. By making use of the adjacency list this step again takes time $\mathcal{O}(n)$. This leads in sum to a runtime of $\mathcal{O}(n) \cdot (\mathcal{O}(n) + \mathcal{O}(n)) = \mathcal{O}(n^2)$ for the construction of one local clique, hence, to a runtime of $\mathcal{O}(n^3)$ for the construction of all local cliques. In the case of processing on the product graph which has a size of $\mathcal{O}(n^2)$ the runtime finally becomes $\mathcal{O}(n^6)$.

Unfortunately merging of local cliques has exponential complexity, since the number of graphs in the set C can become very large, theoretically up to $\sum_{i=3}^{n} \binom{n}{i} = 2^n - \sum_{i=0}^{2} \binom{n}{i} = \mathcal{O}(2^n)$. However, thanks to the thresholds ω and γ', the true number is typically much smaller than this theoretical bound, and most of the time, the set C is already empty after a few iterations of the merging step.

The theoretical space complexity of this approach is not reduced in comparison to the original approach based on the (Bron and Kerbosch, 1973) algorithm, since in the worst case both algorithms consider the power set of the set of nodes in the product graph. The experimental study, however, will show, that such cases will not occur in practice, since the parameters ω and γ_1 lead to a strong reduction of the subsets which must be considered.

6.3 Graph Alignment

Pairwise partial alignments are sometimes insufficient, e.g., if one is interested in the whole structure of a protein binding site. A prime example is the *multiple* sequence alignment used to identify common patterns in a set of amino acids. The concept of multiple graph alignment has been introduced in (Weskamp et al., 2007) as a structural counterpart to multiple sequence alignment. Compared to the graph-based approach presented before, graph alignment calculates a complete alignment, i.e. each element of a graph is assigned to exactly one element in another graph, and vice versa. Moreover it is able to extend pairwise alignments to multiple alignments using an appropriate aggregation technique. Such multiple alignments can be used to analyze a whole set of protein binding sites on the structural level. Since each alignment can be used to calculate a similarity, graph alignment also serves as a similarity measure.

A main drawback of the method proposed by (Weskamp et al., 2007) is that it employs a number of greedy heuristics, none of which can guarantee a certain degree of optimality. To calculate a pairwise alignment, the clique

approach is used in a first step to find the maximum common subgraph, thus a partial alignment, that is used as a seed-solution for the complete pairwise alignment. This step is realized by the Bron-Kerbosch algorithm (Bron and Kerbosch, 1973) which is exact, yet at the cost of an exponential runtime. Nodes, therefore indirectly edges, that are not matched in this first step are iteratively added to the partial alignment in a greedy way until the complete alignment is formed. Hence this approach combines both the high runtime of exact methods solving NP-hard problems and the non-optimality of greedy heuristics applied on non-matriods. Moreover, multiple alignments are calculated using another heuristic, namely star-alignment.

Due to the greedy nature of this approach, this method is likely to miss the optimal solution. To investigate the quality of the solutions generated by the greedy heuristic, an alternative method based on evolutionary algorithms is developed here. In the case of an improvement, the novel method can become a good alternative for calculating alignments of higher quality. Before introducing this novel approach, a brief introduction to multiple graph alignment and its scoring function is given, both taken from (Weskamp et al., 2007).

6.3.1 Multiple Graph Alignment

The problem of structural alignments has already been discussed in a general form in Section 2.5. In this chapter, structures are represented in the form of graphs $G = (V, E, \ell_V, \ell_E)$ and one is looking for multiple alignments of graphs. To consider graphs, the Definition 2.17 must be adapted slightly, by considering the sets of nodes $\mathcal{V} = \{V_1, \ldots, V_m\}$ as a set of objects $\mathcal{X}$. At first sight, this definition does not take the sets $\{E_1, \ldots, E_m\}$ into account. However, since the set E_i $(i = 1, \ldots, m)$ is defined by (a subset of) V_i^2, one-to-one correspondences between edges can be derived indirectly from one-to-one correspondences between nodes. The number of valid alignments thus obtained does not change compared to the general case: There are still $\mathcal{O}(((|V_1| + \cdots + |V_m|)!)^{m-1})$ valid alignments. Hence the goal is to find the optimal multiple graph alignment given a certain evaluation function. Here node-labeled and edge-weighted graphs are considered, therefore an optimal alignment is an alignment that maps such nodes onto each other that share the same label and such edges that share equal weight, and that comes without the insertion of dummy nodes. Of course, one cannot expect to have an alignment in which all labels match and

in which no dummys are inserted for non-identical graphs. Therefore, to be able to construct an alignment, mismatches and insertions of dummys must be allowed. Mismatches and insertions can be considered as edit operations to transform a graph into another graph. By assigning costs to each of these edit operations, it becomes possible to define an edit distance for a pair of graphs G and G'. This distance is defined as the value of the cost minimal sequence of edit operations that transform G into G', hence it can be used to measure the quality of an alignment, since each alignment can be used to derive an edit sequence. The concept of an edit distance can easily be extended to the multiple case, in which a set of m graphs $\{G_1, G_2, \ldots, G_m\}$ is given. In the general multiple case, the goal is to find an edit sequence and a new graph G_{new} so that the m input graphs can be transformed into G_{new} in a cost-minimal way.

The graph edit distance is especially appropriate for the comparison of protein binding sites, since when comparing protein cavities on a structural level, one has to deal with the same mutations that also occur on the sequence level. Corresponding mutations, in conjunction with conformational variability, strongly affect the spatial structure of a binding site as well as its physicochemical properties and, therefore, its graph descriptor. Weskamp et al. (2007) consider the following types of edit operations between a node-labeled and edge-weighted graph $G = (V, E, \ell_V, \ell_E)$ and another graph:

1. Insertion or deletion of a node $v \in V$,
2. change of the label $\ell_V(v)$ of a node $v \in V$, and
3. change of the weight $\ell_E(e)$ of an edge $e \in E$.

Allowing these edit operations, deletions and insertions of pseudocenters caused by mutation in sequence space or conformational difference that affects the exposure of a functional group toward the binding site can be handled. They also allow one to deal with the change if a mutation replaces a certain functional group by another type of group at the same position, or a change of the distance between two pseudocenters due to conformational differences or noise in the measurement.

To measure the quality of a certain alignment therefore it makes sense to make use of the graph edit distance and to search for the alignment that comes with the lowest edit costs. To assess the costs, an appropriate scoring scheme must be defined.

Scoring

The scoring function corresponds to the edit distance mentioned above and follows a sum-of-pairs scheme, i.e., the score s of a multiple alignment $\mathcal{A} = (a^1, \ldots, a^n)$ is defined by the sum of scores of all induced pairwise alignments. Moreover, it consists of two parts, a node score and an edge score that are aggregated by addition, thus the score of an alignment is given by:

$$s(\mathcal{A}) = \sum_{i=1}^{n} \mathrm{ns}(a^i) + \sum_{1 \le i < j \le n} \mathrm{es}(a^i, a^j) \ . \tag{6.10}$$

The node score (ns) is defined by the sum of the evaluations of the individual assignments a^i of the alignment, where such an evaluation is defined as follows:

$$\mathrm{ns}\begin{pmatrix} a_1^i \\ \vdots \\ a_m^i \end{pmatrix} = \sum_{1 \le j < k \le m} \begin{cases} \mathrm{ns}_m & \ell_V(a_j^i) = \ell_V(a_k^i) \\ \mathrm{ns}_{mm} & \ell_V(a_j^i) \neq \ell_V(a_k^i) \\ \mathrm{ns}_{dummy} & a_j^i = \perp, a_k^i \neq \perp \\ \mathrm{ns}_{dummy} & a_j^i \neq \perp, a_k^i = \perp \\ 0 & \text{otherwise} \end{cases} \tag{6.11}$$

Thus, to evaluate a single vector of mutually assigned nodes, these nodes are considered in a pairwise manner, and three cases are distinguished:

1. The labels of two nodes are equal (match),

2. the labels differ (mismatch),

3. one of the nodes is a dummy.

For each case, a parameter ns_m, ns_{mm} and ns_{dummy} is used, respectively, to reward matches or to penalize mismatches.

Comparing two edges is somewhat more difficult than comparing two nodes, as one cannot expect to observe edges of exactly the same length. Two edges are considered as a match if their respective lengths, a and b, differ by at most a given threshold ϵ, and as a mismatch otherwise. The edge score (es) is

then given by

$$\text{es}\left(\begin{pmatrix} a_1^i \\ \vdots \\ a_m^i \end{pmatrix}, \begin{pmatrix} a_1^j \\ \vdots \\ a_m^j \end{pmatrix}\right) = \sum_{1 \le k < l \le m} \begin{cases} \text{es}_{mm} & (a_k^i, a_k^j) \in E_k,\ (a_l^i, a_l^j) \notin E_l \\ \text{es}_{mm} & (a_k^i, a_k^j) \notin E_k,\ (a_l^i, a_l^j) \in E_l \\ \text{es}_m & d_{k,l}^{i,j} \le \epsilon \\ \text{es}_{mm} & d_{k,l}^{i,j} > \epsilon \end{cases} \tag{6.12}$$

where $d_{k,l}^{i,j} = |\ell_E(a_k^i, a_k^j) - \ell_E(a_l^i, a_l^j)|$. Again, constants es_m and es_{mm} are used to reward or penalize matches or mismatches.

6.3.2 Evolutionary Algorithm for Solving the Multiple Graph Alignment Problem

It has already been discussed that the method used so far to solve the multiple alignment problem comes with several problems. First of all, it employs greedy heuristics that do not even guarantee a certain degree of optimality. Secondly, it solves the problem of finding a seed solution by applying clique-detection on the product graph. This step, however, consumes an enormous amount of memory, thus fails even on medium-sized graphs, moreover it is inefficient.

A simple idea to solve these problems is to use an evolutionary algorithm that is introduced in this section, called GAVEO (short for Graph Alignment via Evolutionary Optimization). Even though evolutionary algorithms are also heuristics, they are known to be powerful optimizers and might return much better results than the greedy heuristic mentioned above, albeit at the cost of an even higher runtime. Here, the basic loop and the selection operators of *evolution strategies* are used since they have advantages if applied in very large search spaces as in the case of graph alignments, i.e., they use deterministic selection and ensure that good solutions are not discarded. Since evolutionary strategies are restricted to real-valued optimization problems[2], the representation of individuals must be changes to enable an application on the graph alignment problem. Therefore the genetic operators, namely mutation and recombination must be adapted, too. In the following, the changes are described in more detail.

[2]Meanwhile evolutionary strategies can also be applied on integer optimization problems by exchanging the Gaussian distribution with a binomial distribution.

A:	1	2	$\perp$	4	$\perp$	3	5	6	$\perp$	$\perp$
B:	6	1	2	3	4	5	7	9	8	$\perp$
C:	4	$\perp$	3	2	$\perp$	1	$\perp$	$\perp$	5	$\perp$
D:	$\perp$	3	4	$\perp$	2	1	7	5	6	$\perp$

Figure 6.5: Matrix representation of a multiple graph alignment. Dummys are represented by a $\perp$. Note that the order of the columns is arbitrary.

Representation of Individuals

In evolutionary algorithms, individuals correspond to potential solutions of the problem considered, accordingly individuals represent here multiple alignments. Given a fixed numbering of the nodes of graph G_i from 1 to $|V_i|$, a multiple graph alignment can be represented in an unique way by a two-dimensional matrix, where the rows correspond to the graphs and the columns to the aligned nodes of these graphs or possibly dummys. Figure 6.5 shows an example of such a matrix for the case of 4 graphs of size 6, 9, 5, and 7, respectively. The first column indicates a mutual assignment of the first node of graph A, the sixth node of graph B, and the fourth node of graph C, while there is no matching partner other than a dummy indicated by a $\perp$ in graph D.

The number of rows in an individual is known a-priori, since it corresponds to the number of graphs to be aligned. The optimal number of columns, however, is a-priori unknown. It ranges between the two extremes: $\max_{i=1,\ldots,m} |V_i|$ and $|V_1| + \ldots + |V_m|$. On the one hand, using the upper bound will usually be too large a number and may come along with an excessive increase of the runtime needed to solve the multiple graph alignment problem. From an optimization point of view, a small number of columns is hence preferable. On the other hand, however, using the lower bound flexibility is lost and the optimal solution is excluded with high probability, since only a small part of the search space is considered during the evolutionary search. Generally, it is quite difficult to find a trade-off between both extremes.

Therefore, to avoid this problem, a self-adaptation technique is employed. To this end an adaptive representation is used that does not require the a-priori specification of the number of columns. The matrix scheme is initialized with m rows and $n_{max} + 1$ columns, where $n_{max} = \max_{i=1,\ldots,m} |V_i|$. Hence, large

parts of the search space are neglected but can be added into the consideration according to an update rule: In randomly chosen intervals, it is checked whether further dummy columns are needed or existing ones have become unnecessary. To this end, all individuals in the population are considered and the number of dummy columns is determined. Three cases can occur:

1. In all individuals of the population at least one dummy column exists and in at least one individual exactly one dummy column, which means that the current length is still optimal.

2. All individuals have more than one dummy column: Apparently, a number of dummy columns are obsolete and can be removed, retaining at least one dummy column in all individuals and exactly one dummy column in at least one individual.

3. At least one individual has no dummy column left: The dummy column has been "consumed" by mapping dummys to real nodes. Therefore, a new dummy column has to be inserted in all individuals of the population.

This self-adaptation step is applied on the whole population, since especially the recombination operator requires equal dimensionality of all individuals in the population. As a result, this adaptation technique allows one to decrease the runtime dramatically. Obviously, in most cases it is not necessary to consider the upper bound to find the optimal alignment, therefore starting with a small size and increasing it if necessary leads to the pruning of large parts of the search space, thus to a more efficient search.

The efficiency of an evolutionary algorithm can be increased further by storing the fitness in the individuals to avoid multiple evaluation of individuals. Therefore, individuals are extended by an additional real-number representing the fitness of the individual.

Moreover, a self-adaptation technique (Beyer and Schwefel, 2002) for the step sizes of the mutation operator is applied, allowing a simpler adjustment of the mutation strength, a procedure that is introduced later together with the related mutation operator. Here, it is sufficient to know that the individual is extended by a further integer to allow such an automatic adjustment.

Evolutionary Loop

The evolutionary loop is taken unchanged from evolutionary strategies (Beyer and Schwefel, 2002) and depicted in Algorithm 2.1. Its genetic operators *mating selection*, *selection* and *termination criteria* also remain unchanged. Due to the changed representation, the operators recombination, mutation and the fitness evaluation are adapted.

Initialization

The initialization of one individual is performed by first determining the dimensionality (m, n) of the matrix representing the alignment. This dimensionality is given by the number of graphs considered and the number of nodes in the graphs (cf. representation of an alignment). Having initialized the matrix, each row i is filled with a random permutation of length n. Since row i represents graph G_i that per definition has fewer than n nodes, entries $j > |V_i|$ are replaced by dummys. Another interesting technique that can be applied is based on additional knowledge. Here, the solution obtained by the greedy heuristic can be added into the population or alternatively, the local cliques can be used as starting points for the initialization of individuals.

Recombination

The recombination operator is a mapping $I^\rho \longrightarrow I$ that takes the individuals chosen by the mating selection to create an offspring. To recombine the ρ chosen individuals, $\rho - 1$ random numbers r_i, $i = 1, \ldots, \rho - 1$, are generated, where $1 \leq r_1 < r_2 < \ldots < r_{\rho-1} < m$, and an offspring individual is constructed by combining the sub-matrices consisting, respectively, of the rows $\{r_{i-1} + 1, \ldots, r_i\}$ from the i-th parent individual, where $r_0 = 0$ and $r_\rho = m$ by definition. Simply stitching together complete sub-matrices is not possible, however, since the nodes are not ordered in a uniform way. Therefore, in merging step i, the ordering of the r_i-th row is used as a reference.

This procedure is illustrated in Figure 6.6 for the case $\rho = 3$. Three individuals 'Individual 1', 'Individual 2', and 'Individual 3' and two integers r_1 and r_2 in the range $\{1, \ldots, m = 7\}$ are chosen at random. In this example the random integers 2 and 4 were drawn, hence all individuals are split at the rows $r_1 = 2$ and $r_2 = 4$. The resulting blocks are merged into a new individual (offspring). To preserve the ordering, columns are rearranged according to the rows r_1 and

r_2, respectively, whose indices serve as pivot elements: For example, the first framed subcolumn in 'Individual 1' is copied to the offspring, and since the index in the pivot row r_1 is 2, one has to search for the same index in this row in 'Individual 2'. This subcolumn (framed) is also copied into the offspring. This procedure is repeated for all individuals and columns.

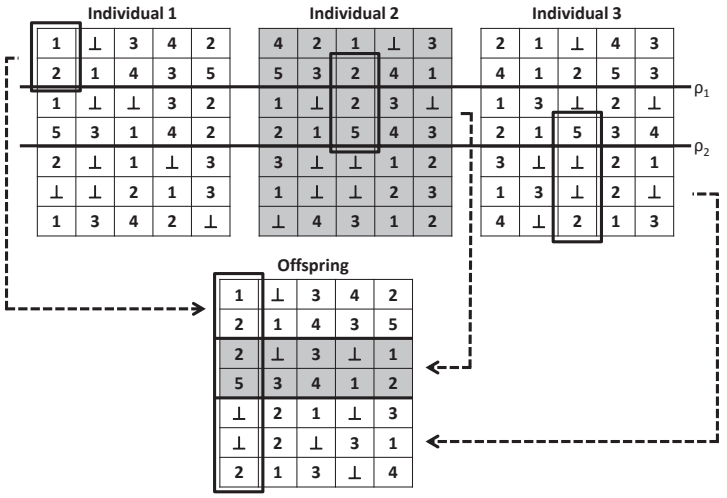


Figure 6.6: Visualization of the recombination operator of the graph alignment via evolutionary optimization approach.

Mutation

The operator mutation : $I \rightarrow I$ selects one row and two columns at random and swaps the entries in the corresponding cells. This procedure obviously reaches small step sizes. To enable large mutation steps, therefore, this procedure is repeated multiple times for each individual. As the optimal number of repetitions was unknown in the design phase of the algorithm, it was specified as a strategy component adjusted by a self-adaptation mechanism (Beyer and Schwefel, 2002).

Here, the self-adaptation technique was realized as follows: The integer stored for this purpose in the individual is mutated first by adding a normally distributed number. After ceiling, again an integer is obtained that specifies the mutation size, that is the number of swaps performed on pairs of randomly chosen cells for each individual that is subject to mutations. Using this parameter allows one to apply, in addition to the simple mutation in which only two cells are swapped, a mutation of much higher impact on the solution. In

particular, self-adaptation allows for an automatic adjustment which does not require human intervention or problem-specific knowledge.

Fitness Function and Acceleration

To calculate the fitness of each individual, the sum-of-pairs measure (6.10) is used here, where dummy columns are of course excluded from scoring, i.e., the insertion or deletion of dummy columns has no influence on fitness. This measure was introduced by Weskamp et al. (2007) to measure the quality of the solutions of their greedy heuristic. Hence, this measure was evaluated $m \cdot (m-1)$ times. In the case of evolutionary optimization, the fitness function is however evaluated many times, thus this measure becomes the bottle-neck of the whole approach. Therefore, a modification of (6.10) is used that still leads to the same mapping.

The naive implementation of the computation of the sum-of-pairs fitness function (6.10) comes with a complexity $\mathcal{O}(n^2 \cdot m^2)$, since the number of summands is $n \cdot m^2$ and $n^2 \cdot m^2$ in (6.11) and (6.12), respectively. Theoretically of the same complexity, the runtime however can be reduced considerably in practice by using information about the distribution of edge labels. To this end, all pairs of columns in the matrix scheme are considered, each of which specifies a set of edges mapped onto each other. Edge weights are sorted in ascending order in time $\mathcal{O}(m \log(m))$, where dummy edges are assumed to have weight ∞. The sorted vector of weights allows the identification of the index i_d from which on dummy-edges follow. Thus, the edge score (6.12) must be evaluated until i_d is reached. The remaining summands can be calculated by evaluating $(m - i_d) \cdot i_d \cdot es_{mm} + (m - i_d) \cdot (m - i_d - 1) \cdot es_m$. The theoretical runtime remains the same, however, since the graphs are rather sparse due to their construction[3], this procedure leads in practice to a considerable win on efficiency.

Complexity

The complexity of the GAVEO approach is clearly dominated by the evaluation of fitness function (6.10) which has complexity $\mathcal{O}(n^2 \cdot m^2)$. Although the complexity of the fitness evaluation is known, the overall complexity of GAVEO cannot be determined due to the reasons mentioned already in Section 4.1. Therefore, the time complexity of GAVEO is again given as a function

[3] if one follows the recommendation of Weskamp et al. (2007) and specifies $\delta = 11$ Å

over l, where l is the random variable specifying the number of iterations the algorithm must perform until the termination criterion holds. The resulting complexity hence becomes $l \cdot \mathcal{O}(n^2 \cdot m^2)$.

The space complexity of this approach is quite low: Since no complex calculations are performed, in particular calculations which require the product graph, GAVEO's space complexity is given by the matrix needed to represent a potential solution of the MGA problem. In the worst case, this matrix has a size of $(m \times nm)$.

6.3.3 Combining Evolutionary Optimization and Pairwise Decomposition

As shown in Definition 2.17, the search space of the multiple graph alignment problem has a size of $\mathcal{O}(((|V_1| + \cdots + |V_m|)!)^{m-1})$, hence it grows super-exponentially with the number of graphs, which is of course problematic from an optimization point of view. Obviously, the efficiency of the evolutionary algorithm depends on the size of the search space, hence efficiency can be increased by down-scaling the search space. One established strategy to reduce the search space is to use decomposition techniques, where the optimal multiple alignment problem is decomposed into a set of pairwise alignment problems. Subsequent, after solving these presumably simpler pairwise problems by means of GAVEO, composition techniques are applied to merge the pairwise alignments to a multiple alignment. Hence, star-alignment can be used for the purpose of a reduction of the search space. In comparison to techniques as tree-alignment or the m-partite pivot graph, star-alignment can be applied in a straightforward way since there is no need to define the costs for assigning two nodes which is not trivial[4]. Decomposing a multiple alignment problem into a set of pairwise alignment problems by using star-alignment was already applied in (Weskamp et al., 2007), however mainly not with the goal of reducing the search space rather than making a calculation possible, since no techniques could be applied to find a seed solution for the multiple graph alignment problem. Here this approach is mainly used to reduce the size of the search space, leading to the GAVEO* algorithm.

[4] Using only the node score is not sufficient since all information about the geometry would get lost.

Complexity

Due to the reduced search space, the runtime of GAVEO* is much lower in comparison to GAVEO. Since only pairwise alignments are considered (leading to $m = 2$), the time needed to evaluate fitness (6.10) is reduced to $\mathcal{O}(n^2)$. Moreover, since the problem becomes much easier to solve, the random variable l, giving the number of fitness evaluations needed until the termination criterion holds, will also decrease strongly. Even though there are $\mathcal{O}(m^2)$ pairwise calculations to perform, hence the overall complexity is also $\mathcal{O}(n^2 \cdot m^2) \cdot l$, the random variable l takes much smaller values and leads to a much more efficient approach. The complexity of the star-alignment procedure is very low in comparison to the evolutionary algorithm and neglected here.

The space complexity of this approach is unchanged in comparison to the original GAVEO method, since the m matrices representing an alignment have a size of $(2 \times n)$, that can grow by merging up to $\mathcal{O}(n^2 \cdot m)$. However, for GAVEO as well as GAVEO*, the space complexity is very low and does not pose a problem for modern computers.

6.4 Summary

Graphs are widely used data structures for which many algorithms exist that can be directly applied on node-labeled and edge-weighted graphs, such as algorithms to calculate shortest paths or concepts like the product graph. More importantly, graphs are a natural representation of relations between objects which makes them very flexible and allows to use them on a variety of applications. Especially in combination with flexible measures such as the graph edit distance, the resulting approaches provide an enormous degree of flexibility allowing for deformations of various kind, e.g. local transformation. This high degree of flexibility is reached by representing an object consisting of a finite set of n elements by $n \cdot (n-1)/2$ distances in a graph-based representation. Obviously, such a representation requires much more space than a representation in the form of coordinates. Moreover a larger set of degrees of freedom may not necessarily be beneficial since the resulting problems become computationally much harder.

In this chapter different ways to measure similarity or distance between graphs were considered. A very simple though efficient method is based on

the R-convolution framework. This framework allows to measure similarity between graphs by decomposing them into substructures and by applying subsequently an all-versus-all comparison of these substructures – a procedure which has already been successfully applied in different domains. Other, more powerful methods for measuring similarity between structured data follow another concept: Here, a search for correspondences between certain substructures is performed. With methods based on subgraph-isomorphism, techniques are available which search for the maximum common subgraph that can be used to derive similarity. Such techniques are often not error-tolerant, i.e. small differences within two graphs may cause very small common subgraphs, hence lead to a small overall similarity for similar objects that were subject to noise, mutations and structural flexibility. Techniques based on quasi-cliques seem to be a good trade-off between meaningful similarity measure and error-tolerance needed for real-world applications. The efficient construction based on local clique merging seems to be an especially promising way to calculate similarity in an efficient and error-tolerant way. However, even more error-tolerance can be introduced by considering methods based on the graph edit distance. Here, graphs are matched completely however after undergoing a set of transformations that are penalized by certain predefined scores. This technique allows one to calculate complete alignments. Moreover, by using merging techniques, multiple alignments which can be used to identify conserved patterns in a set of protein binding sites can be constructed. Algorithms solving this problem were already introduced in the literature, e.g., based on greedy heuristics which however cannot guarantee optimality. The evolutionary algorithm introduced here surely becomes inefficient for larger inputs. It can however be used on smaller inputs or on preselected structures for which it is important to generate an (almost) optimal multiple alignment.

Part III

Validation

7

Experiments

The experimental section addresses several questions. The most important task is obviously to answer which of the proposed techniques is the most appropriate one to measure similarity (or equivalent distance) between protein binding sites. However, it is not easy to answer this question, since the concept of similarity is rather vague and subjective. Therefore, this question is considered indirectly in the form of a classification experiment.

Before considering classification problems some preliminary tasks must be solved. Many of the algorithms introduced have parameters that must be set in an appropriate way. As will be shown later, the setting of parameters is not a trivial task, in many cases it is even almost impossible since the time needed to set parameters optimally would become too large. Therefore in most cases parameters are taken from the literature. An exception are those parameters that do not influence the similarity measure but instead the behavior of the algorithm. Such parameters can be adapted easily and may lead to a dramatic reduction of runtime.

In another experimental study a more relevant practical question is addressed. CavBase is a database that is primarily designed for searching for similar protein binding sites which has been realized so far by means of the subgraph-isomorphism approach, and is substituted by the novel methods developed in this thesis. Using appropriate quality measures allows one to assess the performance of the different measures on this important task.

In the last experiment the one-to-one correspondences produced and multiple structural alignments are considered. Hence, this experiment is devoted to answering the question as to whether the methods proposed here are appropriate for calculating meaningful multiple alignments. To answer this question

datasets were chosen for which common patterns are already known. Using multiple alignment techniques enables the detection of conserved patterns that can be compared against patterns known to be conserved, to assess the quality of the proposed methods.

Due to the large amount of data and possible high complexities of the algorithms, experiments are performed on clusters of which two are available. The local cluster consists of 87 nodes, each of which provides two 2.0 GHz AMD DualCore Opteron 270® processors with 8 respectively 16 GB memory. Moreover 57 nodes, realized in the form of two 2.4 GHz AMD DualCore Opteron 2216 HE® processors and 16 GB memory are available in this cluster. The other cluster *LOEWE*, located at the Goethe-Universität Frankfurt, consists in total of 20,928 CPU cores plus 778 GPGPU hardware accelerators and provides 56 TB memory.

Some of the datasets, experimental settings and results were already published in (Fober et al., 2009b), (Pfeffer et al., 2009), (Fober and Hüllermeier, 2011), (Fober et al., 2011) and (Fober et al., 2012).

7.1 Overview of Developed Methods

Before the experiments are presented, in Table 7.1 a short overview of the developed methods is given, summarizing the time and space complexity of these algorithms in their most efficient realization. Moreover, this table indicates which kind of calculation is performed: Raw similarity calculation without taking one-to-one correspondences into consideration up to the calculation of complete multiple alignments. In this table and in the following experimental study, histogram-based approaches, simplices, R-convolution kernels based on random walks (RW) and shortest paths (SP) and the labeled point cloud superposition (LPCS) are considered as measures which calculate a raw similarity or distance. Approaches which derive this value from one-to-one correspondences, hence which return a partial alignment, are based here on a clique search on the product graph. Cliques can be detected by the (Bron and Kerbosch, 1973)[1] algorithm (BK), the local clique (LC) approach or the local clique merging (LCM) technique. Finally, iterative graph alignment (IGA) taken from (Weskamp et al., 2007), graph alignment via evolutionary optimization (GAVEO) and geometric alignment (3DA), all belong to the class of algo-

[1] Complexity taken from (Tomita et al., 2006).

Table 7.1: Overview of developed methods used in the experimental study.

METHOD	HISTOGRAM	SIMPLICES	RW	SP
TIME	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^6)$	$\mathcal{O}(n^4)$
SPACE	$d_{\max}$	$N_o(n,k)$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$
PARTIAL	✗	✗	✗	✗
COMPLETE	✗	✗	✗	✗

METHOD	LPCS	BK	LC	LCM
TIME	$\mathcal{O}(l \cdot n^2)$	$\mathcal{O}(3^{\sqrt[3]{n^2}})$	$\mathcal{O}(n^6)$	$\mathcal{O}(2^n)$
SPACE	$\mathcal{O}(n)$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$
PARTIAL	✗	✔	✔	✔
COMPLETE	✗	✗	✗	✗

METHOD	GAVEO	IGA	3DA
TIME	$\mathcal{O}(l \cdot n^2 \cdot m^2)$	$\mathcal{O}(m^2 \cdot 3^{\sqrt[3]{n^2}})$	$\mathcal{O}(m \cdot n^3)$
SPACE	$\mathcal{O}(n \cdot m)$	$\mathcal{O}(2^n)$	$\mathcal{O}(n^3)$
PARTIAL	✔	✔	✔
COMPLETE	✔	✔	✔

rithms constructing complete multiple alignments. Given a set of m protein binding sites, the variable n gives the number of pseudocenters appearing in the largest protein binding site and the variable $d_{\max}$ the largest distance between two pseudocenters. For methods based on evolutionary algorithms the runtime cannot be estimated easily. Here the costs of one fitness evaluation are given that however must be multiplied with a variable l giving the number of performed fitness function evaluations. Hence, by specifying a fixed number of iterations the runtime can be estimated. In practice, however, more sophisticated termination criteria are used. Hence l becomes a random variable.

7.2 Parameter Settings

Most of the algorithms introduced have several parameters allowing an adjustment of the algorithm onto a certain application. A drawback of such parame-

ters, however, is that an inappropriate setting can lead to a poor performance of the algorithm. Moreover, the determination of a parameterization is often difficult and requires time, often making the overall approach inefficient. Here, two different parameterizations are considered: Those which adapt the algorithm, e.g. the population size of an evolutionary algorithm, and others that influence the calculated similarities, e.g. edit-costs of graph edit distance or the bin-size of feature-based approaches. Where for the former problem often one representative input is sufficient, the latter problem requires the specification of a benchmark set which is used to estimate if a certain parameterization performs well, determined e.g. by using a time consuming cross-validation. Here, different problems appear: The benchmark set should be representative for the whole CavBase, moreover, parameters optimized on this set should generalize as much as possible. Obviously, the latter problem is much harder, a reason why in different works (e.g. (Weskamp et al., 2007)) reasonable values were chosen for such parameterizations. Therefore, in this thesis only the parameterization of algorithms is considered comprehensively, in particular because a parameterization not optimally chosen can lead to an increased runtime. In combination with the large database this can become a serious problem.

For parameterizing an algorithm a goal must be formulated. To realize a fast response of the database the runtime of algorithms must be minimized. Hence, the goal is the minimization of runtime, a task which is tackled on a representative input. Different concepts can be used to determine optimal parameters. If parameters are independent, methods based on the one-factor-at-a-time approach can be used that allow the adjustment of each parameter separately. Generally, most of the parameters considered will be mutually dependent, thus a method which takes correlations into account must be applied. Design of Experiments (Anderson and Whitcomb, 1974), e.g., provides a set of methods, as partial designs, factorial designs and central composite designs, which can be used to determine optimal parameters experimentally. Here, a novel approach introduced by Bartz-Beielstein et al. (2005) will be used, which is described in the following.

7.2.1 Sequential Parameter Optimization Toolbox

The sequential parameter optimization toolbox (SPOT) (Bartz-Beielstein et al., 2005) was developed for the optimization of a set of parameters using com-

puter experiments. SPOT is a semi-automatic method that tries to keep the computational costs for determining an improved parameterization low by internally fitting a model Y that is able to predict the quality of the algorithm in terms of the given quality criterion for a certain so far untested parameterization of the algorithm. For this it uses a regression model with polynomials of order 2 and a Gaussian correlation function. Assuming that the algorithm has k exogenous parameters this response requires to fit $q = (k^2 + 3 \cdot k + 2)/2$ variables (Sacks et al., 1989), thus requires to have at least q parameterizations of the algorithm with dependent output values. In a first step, these q points have to be chosen, a task for which the user has to define a set of predefined intervals, each specifying the values allowed for a parameter (e.g. $[1, 50]$ for the population size) called the region of interest (ROI). After defining the ROI, a set of parameters is generated to be used in the algorithm. In the beginning of this procedure, latin hypercube sampling (Iman, 2008) is performed to generate the required q parameterizations. This sampling procedure divides the hypercube defined by the ROI into a set of sub-hypercubes from each of which a predefined number of parameterizations is sampled. This has the benefit that it is guaranteed that the design points are scattered uniformly over the whole ROI compared to a simple sampling scheme. After creating the initial parameterizations, the SPOT-loop illustrated in Figure 7.1 is executed. In each iteration, the chosen parameterizations are evaluated by running the algorithm

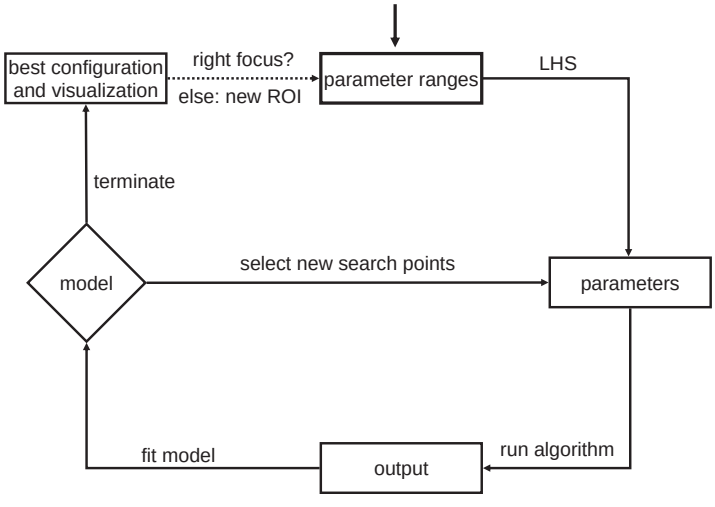


Figure 7.1: Loop of the sequential parameter optimization procedure.

with this set of values obtaining another value (e.g. runtime) giving the quality of the considered parametrization. Subsequently, SPOT allows the building of a regression model using the derived parameterizations and the corresponding quality values. The model obtained is further used to generate additional parameterizations x for the following purposes:

1. Improving the regression model, and

2. finding better parameterizations.

The second (and most important) purpose is fulfilled, once the improvement (7.1) is maximized for x:

$$I(x) = \begin{cases} f(x) - f^* & \text{if } f^* > f(x) \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

where f^* denotes the best value found so far. However, the exact value $I(x)$ will not be known a priori, so that the SPOT has to use the expected improvement based on the model Y. This procedure is continued until a certain termination criteria holds, usually the number of loops performed.

Optimizing GAVEO Parametrization

The identification of a representative input for the evolutionary algorithm solving the graph alignment problem is not easy. Even though the optimization problem considered is a combinatorial, (probably) multimodal problem for all inputs, the search space grows exponentially with the size of the input. Generally, the search space may influence the optimal parameterization, hence SPOT is applied on inputs of different sizes, where for each size equal structures are used, which on the one hand ensures that the optimal alignment hence the optimal score is known in advance. Knowing the optimal solution on the other hand allows the determination of time needed until the algorithm reaches the optimal solution. Hence, for a certain parameterization the evolutionary algorithm is called and the time needed to reach the optimum is recorded. To catch parameterizations leading to a very high runtime the algorithm is terminated if the optimum could not be reached within 120 seconds. Then the run is considered as unsuccessful and the time 180 seconds is recorded. The Table 7.2 gives the optimal parameters determined after 25 SPOT loops, where in each

iteration two new parameterizations were generated and tested. The structures used for small (s), medium (m) and large (l) inputs were chosen as follows[2]:

small: `3FP9.16`, which is a binding site containing 16 nodes and 102 edges using 11 Å as cut-off distance.

medium: `1TF8.1`, that contains 54 nodes and 858 edges for the same cut-off value.

large: `2GVN.11`, which has 142 nodes and 3265 distances below 11 Å.

Table 7.2: Optimal exogenous parameters for GAVEO found by SPOT for optimizing graph alignments of different sizes.

PARAMETER	μ	ν	ρ	κ	stall generations	adaptation probability
VALUE (S)	4	3.72	2	281	255	—
VALUE (M)	5	0.80	2	2160	1550	—
VALUE (L)	4	3.88	2	1580	1632	—
VALUE (FINAL)	4	4	2	∞	—	0.35

Obviously, similar parameterizations were determined independently of the search space size. Even though the parameter ν was chosen small for the case of medium inputs, there were parameterizations with almost the same runtime in which ν was set around 4. Therefore, the use of a general parameterization given in the last row of Table 7.2 seems to be justified. Effect plots in Figure 7.2 indicated the enormous influence of different parameter settings on the runtime, moreover these plots visualize the ROI used in SPOT. It is indeed astonishing that a small population size $\mu = 4$ is sufficient to find the optimal solution. This indicates clearly that the problem of local optima does not appear during optimization and that a small population, large enough to enable recombination, is sufficient. The recombination parameter ρ was chosen as the largest possible value, namely 2. Due to the simple realization of mutation, recombination becomes an important operator for the optimization, therefore, it should be enabled as often recommended in the literature (Schwefel, 1993). The values for κ were chosen much higher than the generations required for

[2]Other inputs were also tested, leading to similar results.

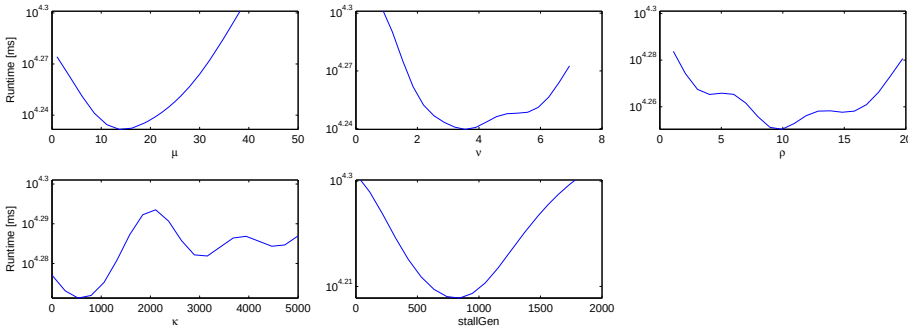


Figure 7.2: Effect plots (logarithmic y-axis) for the different evolutionary algorithm parameters. Obviously, the runtime strongly depends on the chosen parametrization.

reaching the optimal solution, indicating that a plus-selection should be applied, which sounds reasonable for search spaces of this size. Beside the exogenous parameters of the evolutionary algorithm the termination criterion *stall generations* was evaluated. If this parameter is set to too small a value the optimization will be not successful, since the optimization is terminated before reaching the minimum. Using the other extreme in which too many generations without improvement are used until the procedure is terminated will lead to an artificially high runtime. Here, the experiments indicated that this parameter depends on the size of the search space, but that the number converges to a value of about 1600. The parameter giving the probability for length adaptation could not be adjusted in this procedure since for identical inputs no dummys are needed, hence the length adaptation becomes unnecessary. This parameter however was adjusted in a similar way, where for unequal graphs and different settings of the probability parameter the fitness of the solution was recorded that was obtained after a fixed number of fitness function evaluations. This procedure to adjust parameters obviously requires independence of the length adaptation parameter and the other parameters already adapted (as population size and selective pressure). However, it was shown in Table 7.2 that parameters are independent of the size of the inputs. The size of the input obviously correlates with the size of the alignment. Hence, one can assume independence between the parameters adapted in Table 7.2 and the parameter giving the probability for length adaptation. The value thus obtained for the length adaptation probability is 0.35. This probability seems very high, however, it avoids long times of stagnation of the algorithm, e.g. if there are no

possibilities of improving the solution found so far due to an inappropriate alignment length. Moreover, in a separate test-procedure the self-adaptation mechanism for mutation was tested. Here it was clearly seen that the self-adaptation mechanism decreases each setting of the parameter σ to 1 within the first generations. This indicates that the self-adaptation mechanism introduced clearly works properly, but that larger mutation steps as defined here do not prove beneficial. This result seems reasonable, since in search spaces of this size a successful swap of two cells will usually be followed by an unsuccessful swap (i.e. a swap destroying the fitness improvement obtained by the first swap). Therefore, in the following self-adaptation of the mutation strength is no longer considered and mutation is defined as an operator which performs exactly one swap of two cells in the matrix representing the alignment.

Optimizing LPCS Parametrization

The representative input of the evolutionary strategy solving the optimization problem appearing in the labeled point cloud superposition approach is an arbitrary point cloud, since each point cloud leads in the general case to the same optimization problem: A multimodal function with domain $\mathbb{R}^3 \times [0, 2\pi]^3$ and range $[0, 1]$. For solving this kind of problem the state-of-the-art evolutionary strategy is used that has 9 exogenous parameters. The sequential parameter optimization toolbox was applied on different inputs all leading to similar results. Here the results obtained for the input (`1H2A.4`, `1H2A.4`) are presented. The optimal solution for this input is obviously $\mathbf{0} = (0, 0, 0, \pi, \pi, \pi)$ with corresponding fitness value 1.0. The time needed to reach the minimum is recorded. If this minimum could not be reached within 20 seconds the run is considered unsuccessful and a time of 60 seconds is assumed. The SPOT loop was executed 25 times, where two new designs were evaluated per iteration. At

Table 7.3: Optimal exogenous parameters for the evolutionary strategy found by SPOT for optimizing superpositions of labeled point clouds.

PARAMETER	μ	ν	ρ	σ	rec_x	rec_σ	c_τ	κ	n_σ
VALUE	27	5.17	16	6.15	`i`	`d`	1.18	3408	`true`

the end of the process the optimal parametrization given in Table 7.3 is determined. As expected for a multimodal problem, the population size and the selective pressure were chosen as high values, $\kappa = 3408$ indicates that the plus-

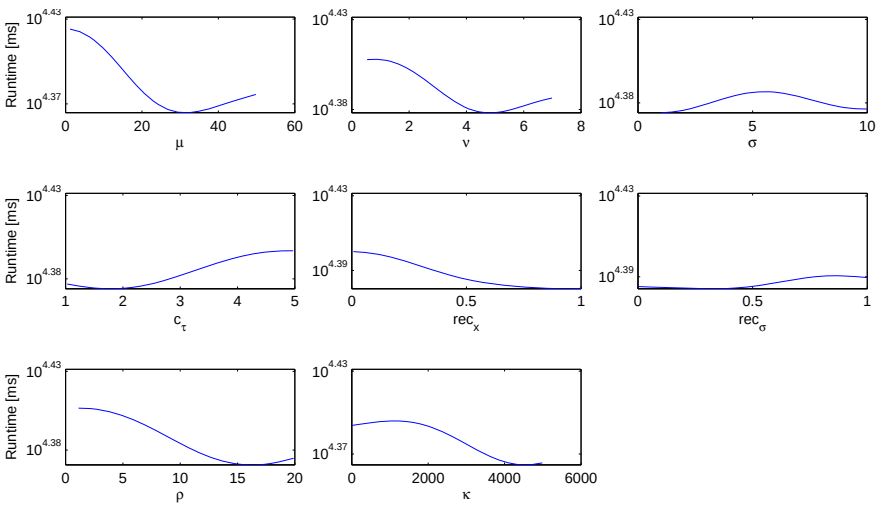
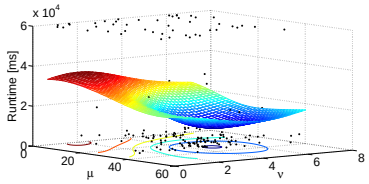


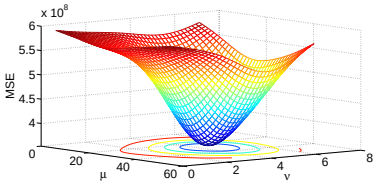
Figure 7.3: Effect plots (logarithmic y-axis) for the different evolutionary strategy parameters. Obviously, the runtime strongly depends on the chosen parametrization.

selection is the best choice for optimization, since the number of generations required is much smaller than 3000. The start step size used for translation[3] was set to $(6.15, 6.15, 6.15)$ and reflects the fact that protein binding sites differ strongly in origin which makes a larger translation necessary. For recombination of the strategy component discrete recombination is used which is able to hold values more stable than intermediate recombination does. This probably suppresses the trend that step sizes decrease too fast. The high influence of the parametrization is illustrated in Figure 7.3 together with a visualization of the ROI. Especially for the parameters μ and ν the influence on the algorithm is quite high. This is additionally illustrated impressively in Figure 7.4 where the response of different (μ, ν) pairs is plotted. For some pairs (μ, ν) the evolutionary strategy was not successful within 20 seconds, other parameterizations performed much better and led to a runtime significantly under 3 seconds. The parameter deciding if n step sizes should be used was not evaluated within SPOT. Instead separate tests were performed, which clearly indicated that n step sizes should be applied instead of 1 step size. This is reasonable since the rotation angles are bounded whereas the translation vectors are not.

[3]The search space for rotation is $[0, 2\pi]^3$, hence (π, π, π) was chosen as a reasonable value without optimizing it.



(a) Influence of the parameters μ and ν: Dots represent measured runtimes, the surface the fitted model



(b) MSE of the fitted model

Figure 7.4: Evaluation of the parameters μ and ν: Runtimes, fitted model and MSE of the model.

7.3 Classification

Having found optimal parameters for the algorithms LPCS and GAVEO (the remaining algorithms have no parameters influencing the calculations), in the next step the quality of the similarity measures can be evaluated. In this thesis a large set of measures on protein binding sites was collected, some of which that are very efficient, others that are computationally more complex using much more information to derive similarity or distance, as e.g. one-to-one correspondences. Moreover, different concepts for representing protein bindings sites were considered beginning with such concepts using the given raw information in the form of labeled point clouds up to a transformation into a node-labeled and edge-weighted graphs. In this section these measures are analyzed to answer the question as to which of the models representing a protein binding site is the best choice and which method to measure similarity (or distance) performs best. Since the concept of similarity (or distance) is quite vague and subjective, it is however not easy to assess the quality of such measures. Therefore an indirect way is chosen, in which the quality is measured in terms of the result of a classification experiment where the assumption is used, that a certain measure performs well if it leads to a high classification rate. The classifier that is used here is the k-NN classifier (Bishop, 2006) for $k = 1,3,\ldots,9$ which is embedded in a leave-one-out cross validation procedure (Bishop, 2006). Hence, to determine the classification rate of a measure on a dataset of size n, the dataset is split into a set of $(n-1)$ elements used for training and one element used for testing. Taking each element once for the purpose of testing, in sum n classifications are performed leading to the clas-

sification rate which is defined as number of correct classifications divided by the number n. In the case of the k-NN classifier the term *training set* is somehow misleading since no classifier is learned on this set. Instead, the element used for testing is assigned to the most frequent class from the multiset of the classes of the k elements in the training set which have the lowest distance or the highest similarity to the element used for testing. Thus, the k-NN classifier is very sensitive towards the measure used, since small differences in the measure can already lead to a completely different classification, thus this classifier should be most appropriate for the purpose of estimating the quality of a measure.

7.3.1 Dataset

To enable the classification experiment two sets of protein binding sites are compiled, each of which contains binding sites that share respectively adenosine-5'-triphosphate (ATP) or nicotine amide dinucleotide (NADH) as a cofactor. The procedure to generate these sets is as follows: CavBase is used to retrieve all known binding pockets that are co-crystallized with the respective ligand. Subsequently, the so obtained sets are reduced to one cavity per protein, thus representing the enzymes by one single binding pocket.

As protein ligands adopt different conformations due to their structural flexibility, it is likely that the ligands in the data are bound in completely different ways, hence the corresponding binding pocket does not necessarily share much structural similarity. Thus, binding pockets with ligands bound in similar conformation are chosen according to the following procedure. The Kabsch algorithm (Kabsch, 1976) is used to calculate the rmsd between pairs of ligand structures. Subsequently, all proteins whose ligands yielded an rmsd value below a threshold of 0.2 Å are combined, thereby ensuring a certain degree of similarity. This value is chosen as a trade-off between dataset size and similarity. Hence, two sets of 141 ATP-binding proteins and 214 NADH-binding proteins are obtained that can be used, e.g. to define a binary classification problem. Since ATP is a substructure of NADH, and hence the former can possibly bind the same ligands as the latter, this dataset is especially challenging. The concrete realization of both sets is given in the Tables B.1 and B.2 in the appendix of this thesis.

Even though this thesis is focusing on the structural analysis of protein binding sites, in Table 7.4 classification rates are presented for this dataset ob-

Table 7.4: Classification rates on the ATP/NADH dataset obtained by using an k-NN classifier and sequence alignment scores for different values of k.

k	1	3	5	7	9
rate	0.898	0.868	0.865	0.837	0.820

tained by using the aforementioned experimental setting and a sequence alignment method. In detail, the well-known Smith-Waterman algorithm was used along with the Blosum-62 substitution matrix to obtain the required similarities between all pairs from the ATP / NADH dataset.

7.3.2 Investigation of Geometric Approaches

Approaches based on a geometric representation are on the one hand quite rigid, a property which could result in some problems, especially if such measures are applied on noisy data which is subject to structural flexibility and mutations. On the other hand, however, such approaches do not cause a loss of information, scale very well and do not lead to exorbitant large search spaces containing even geometric infeasible solutions. Hence they should be very powerful measures from this perception. The LPCS method as introduced in this thesis comes with a parameter λ influencing the obtained similarity scores. This parameter is evaluated in the following by using a linear (1-dimensional) grid starting in zero and going up to 1 in 0.1 steps. For $\lambda = 0$ a subset relation is considered, whereas for $\lambda = 1$ the concept of equivalence is used to measure similarity (cf. Section 2.4.1). The results summarized in Table 7.5 clearly indicate that LPCS is a very good measure on protein binding sites, at least on the binding sites contained in the ATP / NADH dataset, since classification rates of above 90% could be reached on this non-trivial classification problem. Considering the influence of the parameter λ, it indeed sticks out that this parameter has some influence on the classification rates, as illustrated in Figure 7.5. Even though, the influence of λ on the classification rate is not very high, it turns nevertheless out that a trade-off between equivalence and inclusion performs best. Obviously, this is in particular the case for settings of λ in the interval $[0.5, 0.8]$. This setting has benefits compared to $\lambda = 1.0$ especially in the case of two structures that differ in size. Using strict equivalence similarity is defined by the minimum of the score of two superpositions, however, one of the two calculated superpositions will place the smaller one in the middle of the other

Table 7.5: Classification rates on the ATP/NADH dataset obtained by using LPCS scores for different values of λ and k.

λ \ k	1	3	5	7	9
0.0	0.896	0.873	0.865	0.834	0.825
0.1	0.899	0.873	0.868	0.837	0.828
0.2	0.904	0.879	0.870	0.837	0.831
0.3	0.907	0.878	0.868	0.831	0.831
0.4	0.910	0.885	0.870	0.842	0.825
0.5	0.907	0.893	0.879	0.859	0.856
0.6	0.901	0.901	0.896	0.882	0.854
0.7	0.907	0.910	0.870	0.873	0.848
0.8	**0.913**	0.899	0.865	0.854	0.870
0.9	0.904	0.885	0.879	0.868	0.867
1.0	0.890	0.873	0.854	0.845	0.828

one in order to minimize the sum of distances between equally labeled points. Even though, the sum of distances is minimized, the score this superposition returns is low. Hence, even if the other superposition returns a high score, the overall score will be artificially low. However, compared to majority voting, which would lead to a classification rate of 60.28% on the ATP / NADH dataset, LPCS performs independently of the chosen λ value significantly better.

This behavior needs to be reflected in a more detailed way, in particular because LPCS has the drawback of processing on a very rigid representation in the form of labeled point clouds. Although, the classification rates of the other methods are not known so far, one can easily imagine, that increasing classification rates of more than 90% on a non-trivial classification problem is not simple. To enable some insights into the LPCS measure, Figure 7.6 is used which is giving a representative superposition of two protein binding sites. Due to the rigid representation of protein binding sites one cannot expect a perfect match, however, in contrast to concepts as the maximum common subgraph, LPCS is not requiring perfect matches. Instead, not perfectly matching points are also considered, where their deviations are penalized, reducing the score LPCS returns for this comparison. From this point of view, LPCS can

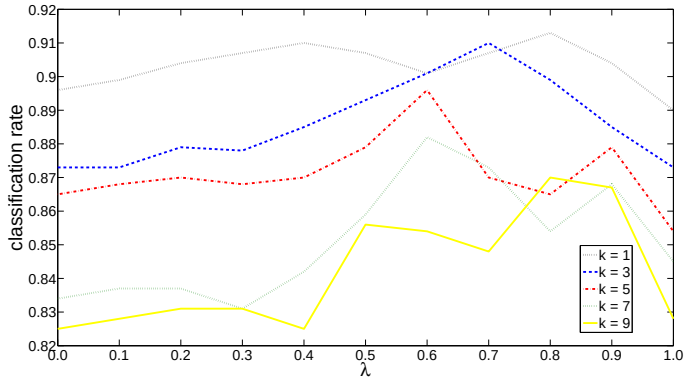


Figure 7.5: Classification rates of k-NN classifier versus λ for different choices of k.

even be considered as a kind of extended rmsd value of two structures, that does not require to have a one-to-one correspondence between points and that even takes physicochemical properties into account. However, theoretically this approach (or more precise the underlying representation) has also a central disadvantage which is illustrated in Figure 7.7, where two sets of points are given. Considering them as point clouds allows no transformation (at least by applying the algorithm presented in this thesis). Hence the LPCS approach tries to minimize the distances between pairs of points from different point clouds leading to the result depicted in Figure 7.7. By allowing to modify the

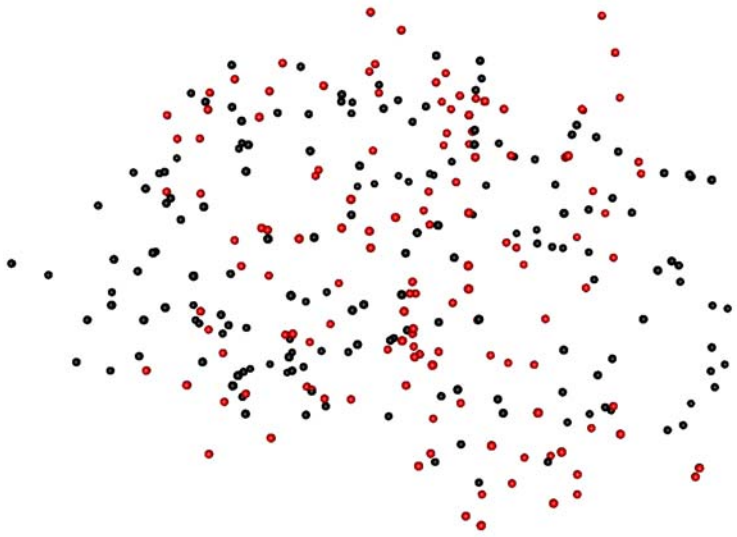


Figure 7.6: Superposition of the structures `1dy3.1` (red) and `1j1z.10` (black).

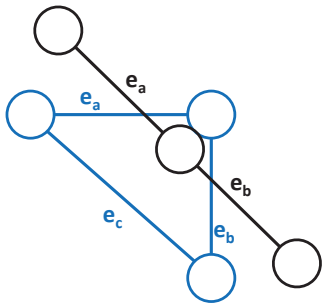


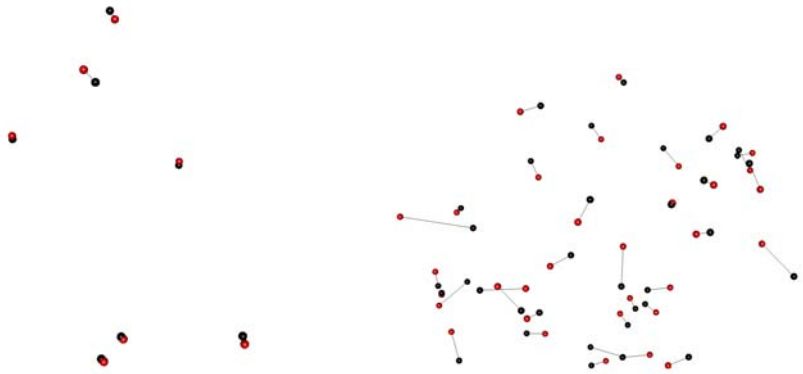
Figure 7.7: Changing the angle between the two lines connecting the outer balls would allow a perfect match; an algorithm working on a rigid representation, however, must accept a mismatch and must try to minimize it.

model (e.g., as in the case of methods following the concept of graph edit distance), a simple adaptation of the angle between the lines connecting the outer nodes would allow for a perfect match. Such an adaptation could be realized by changing edge weight $\ell_E(e_c)$ to $\ell_E(e_a) + \ell_E(e_b)$.

Consideration of the Constructed Alignment

So far it was shown that LPCS scores lead to very high accuracies when applied as inputs for a classifier, even though the underlaying model is very rigid. To explain this behavior further, in the next step the calculated superpositions are inspected closer by transforming them into alignments. Although the similarity scores obtained by LPCS do not depend on the alignments, the alignments strongly depend on the superpositions. Moreover, the conserved pattern of the alignment has the highest contribution to the LPCS score obtained.

Having found the optimal superposition, graph matching (cf. Section 4.2) can be applied to obtain an alignment. Finally, a conserved pattern can be derived from this alignment. Conserved patterns derived from the alignment of the structures `1dy3.1` and `1j1z.10`, both which are taken from the ATP set are visualized in the Figure 7.8 after an additional superposition calculated by means of the Kabsch (1976) algorithm. It is worth mentioning here, that the two structures are not “cherry-picked”, instead the results are representative for many tried structures, hence they will be used throughout the whole study. As can be seen, even a rigid geometric approach is able to superimpose two point clouds in a way that a common pattern of the two clouds is located spatially close. Hence, the distances between pairs of points from the different



(a) Geometric alignment, $k = 0.75$ Å; rmsd $= 0.43$ Å

(b) Geometric alignment, $k = 4.00$ Å; rmsd $= 1.88$ Å

Figure 7.8: Visualization of the conserved pattern ($\omega = 1, \xi = 1$) obtained by constructing geometric alignments of the structures `1dy3.1` (red) and `1j1z.10` (black). One-to-one correspondences are visualized in terms of a straight line.

clouds that participate in this pattern are quite low. As a result, they contribute strongly to the score. On the other hand, points that do not have a counterpart in the other cloud which is spatially close have a very low contribution to the score. Thus, the final score is determined in a similar way as it was done in the mcs measure, namely by dividing the size of the common subgraph by the size of the larger graph. However, in contrast to the classical mcs measure LPCS is taking also the points into consideration which do not match, however, it down-weights them according to the exponential function of the negative distance. In Figure 7.8 alignments are shown in which, respectively, assignments with distance above 0.75 Å or 4 Å are removed. In the alignment depicted in Figure 7.8 (a) indeed a conserved pattern can be recognized which matches almost perfectly. As will be shown later, the largest common pattern that matches up to an error of 0.2 Å has a size of 7 pseudocenters, too. Hence, LPCS was able to superimpose the structures in a way that a pattern of the same size as in the case of the maximum common subgraph approach (see results on graph-based methods) was placed spatially close. Moreover, by increasing the maximally allowed distance between assigned pseudocenters an even larger approximate common pattern is found. Using the assignments to derive an edit path, finally an error-tolerant approach is obtained. However, the error-tolerance is not affecting the structure completely: The transformation rules comprise only of some slight movements, the overall structure of the protein binding sites

remains the same. This is underpinned by the obtained rmsd value which is below 2 Å. Reducing k to 0.75 Å, the rmsd is becoming 0.43 Å and the alignment does not tolerate larger deviations, hence, the number of pseudocenters in the conserved pattern is reduced.

Summary

The results obtained can be summarized as follows: Methods based on point clouds are very interesting tools to process on protein binding sites. It was shown, that the similarity measure LPCS leads to high classification rates. On a non-trivial classification problem as it was used here, a rate of above 90% clearly indicates a very effective measure. By chosing λ-values different from 1, LPCS allows to lessen the concept of equivalence. This may have benefits because the LigSite algorithm used to detect cavities on the surface of proteins has some problems to identify the border of a cavity. Here, a measure based on the trade-off between equivalence and inclusion has clearly advantages which were also indicated by the increased classification rates. A further technique allowing for calculations of alignments is also available for this representation. Among others, such alignments can be considered as a transformation rule to transform the first point cloud into the second one, or vice versa. Hence, this technique introduces at least to some degree flexibility to the representation *labeled point cloud*. The rmsd values obtained when considering conserved patterns behave proportional to the parameter k. Hence, geometric alignment allows the user to adjust the degree of structural flexibility a-priory. On the other hand, this allows the construction of rigid alignments but also of such alignments that are more flexible in terms of deformation of the structures, always by avoiding the complete deformation of the structure.

7.3.3 Investigation of Feature-based Approaches

Feature-based approaches as introduced in this thesis are quite simple and could be a good alternative to measures coming with a higher complexity, as measures based on the graph edit distance. However, it is still an open question if these measures can perform well due to their simplicity, a question that is addressed in this part of the thesis. Therefore, in a first test it is again interesting to see if the measures perform better than majority voting, a procedure which would lead to a classification rate of 60.28% on the ATP / NADH classifi-

caton problem. Another goal of this section is the reduction of a large number of possible realizations of distance, since especially feature-based approaches allow for application of different techniques leading to a combinatorial space, hence to a vast number of possible distances. In the following, all three representations, *seperate handling of properties and distances*, *joint handling of properties and distances* and *simplices* are considered seperately.

Handling Properties and Distances Separately

First of all, the simplest approach introduced in this thesis is considered, namely the histogram approach in which distances and physicochemical properties are considered separately. Using the experimental setting as described above, the results given in Table 7.6 are obtained. In this approach, the comparison

Table 7.6: Results obtained by representing physicochemical properties and distances by histograms, and applying subsequently measures on histograms. On the histograms representing physicochemical properties bin-by-bin measures are applied, whereas cross-bin measures are used to compare distance-histograms. The combination of both histogram types, realized by using the L_2-norm, is given in the last row.

	k	1	3	5	7	9
PROPERTIES	JACCARD	0.758	0.749	0.721	0.732	0.721
	MF ($p = 1$)	**0.783**	0.755	0.729	0.729	0.716
	MF ($p = 2$)	0.724	0.716	0.710	0.707	0.713
	MF ($p = \infty$)	0.710	0.699	0.699	0.693	0.710
DISTANCES	KS	0.589	0.614	0.645	0.665	0.656
	MATCH	0.594	0.623	0.611	0.639	0.654
	EMD	0.665	0.676	0.685	**0.687**	0.673
	QF	0.597	0.555	0.569	0.580	0.625
COMBINED EMD AND MF ($p = 1$)		0.797	**0.837**	0.806	0.806	0.789

of protein structures is reduced on the one hand to the comparison of their respective physicochemical property histograms, using different bin-by-bin distance measures. In light of their simplicity, all variants perform surprisingly well. Moreover, since the number of physicochemical properties is linear in the number of pseudocenters, this approach scales linearly in the size of pro-

tein binding sites, hence it is very efficient. This is, however, only the case if the used distance on histograms does also scale linearly. On the other hand, the results obtained by considering only distance histograms are significantly worse, a phenomenon, which does not necessarily suggest that the geometric structure of a binding site is less important than its physicochemical composition. In fact, the consideration of pairwise distances is a very coarse representation of the geometry of a protein binding site, actually it may even happen that binding sites of different geometry exhibit similar distributions of distances. Therefore, the bad performance of this representation is caused by a very high degree of loss of information. In addition, this approach is computationally more expensive, due to the quadratic number of distances that appear in a protein binding site. It is, however, still quite efficient and comes with a quadratical time complexity, if the complexity of the used measure on histograms is linear in the number of bins.

The histogram approach is mainly designed to be used as a combination of both, the physicochemical properties and the distances. To realize this idea an L_2-norm is used to combine the calculated distances. To avoid a consideration of all 16 possible combination of distances on histograms a preselection of distances is performed. On the physicochemical properties the results are quite similar, independent of the used distance. The only exception is the Minkowski norm with parameter $p = \infty$ which is returning as overall distance the distance between these two bins, whose distance is maximal. Hence, this measure is sensitive towards one certain physicochemical property, namely this property which is maximizing the distance. All other physicochemical properties, even though their distributions match, are taken completely out of consideration. This is obviously an undesired effect and leads to the bad result. Considering distances between pseudocenters, another result is obtained. Here the earth movers distance performs clearly best, where the term “best” is of course misleading since no measure perform really good. In fact, some of these measures even fall below the rate of majority voting. This becomes clear by considering the resulting histograms which have almost all the same shape resulting from the fact that there are only few small and large distances between pseudocenters, whereas the distances in between both extremes appear more often. All these histograms are obviously very similar and do not allow to separate data in a correct way. It seems that the earth movers distance can handle such histograms better than the other measures. The Kolmogorov-Smirnov and the

match distance, e.g., are using cumulative distributions which are making the comparison even more coarse, the quadratic form on the other hand a scoring matrix which is hard to adjust. Due to this reasons, in the combined histogram measure the Minkowski distance ($p = 1$) is used to compare physicochemical property histograms and the earth-mover distance to compare distance histograms. The thus obtained scores are subsequently combined in terms of the L_2-norm.

By using this procedure to select measures independence must be assumed, otherwise possible interactions could lead to a better performance of another combination. In fact, independence is probably not given, however, searching for the best combination would result in an overfitting that is undesired. Moreover, the chosen combination of the Minkowski form, which is known to be a good measure on histograms (coming without ground distance), and the earth movers distance, which is very powerful and interesting from a computational point of view, is also a reasonable choice. The results obtained by applying this combination are given in the last row of Table 7.6, where a classification rate of more than 80% is reported. Hence, the combination of both measures by using the L_2 distance indeed leads to an increased accuracy what in sum is making this measure to an appropriate distance on protein binding sites, which performs significantly better than majority voting and also better than considering one of the two histogram types separately. As a drawback, it is using distance histograms. The construction of these histograms comes with a complexity which is equal to the complexity of the more complex approach in which physicochemical properties and distances are considered jointly. Therefore, before continuing with the approach in which physicochemical properties and distances are regarded separately, in the following the approach is evaluated in which both informations are joined in 28 histograms, since it takes more information into account and does not increase complexity.

Handling Properties and Distances Jointly

Originally, the former histogram approach was designed as a starting point for a more sophisticated and probably more exact measure on protein binding sites. This measure is using 28 histograms instead of two and is combining them in terms of an L_2-norm. The advantage of this representation is that each of these histograms is combining the physicochemical properties and dis-

tances, hence that much more information is available. The results obtained by applying this approach are summarized in Table 7.7 for different measures on histograms. Even though, the histograms used here are considering distances,

Table 7.7: Overview on the results obtained by handling physicochemical properties and distances jointly, and by applying different measures on histograms.

	k	1	3	5	7	9
BIN-BY-BIN	JACCARD	**0.873**	0.870	0.848	0.842	0.834
	MF ($p = 1$)	0.870	0.854	0.794	0.769	0.763
	MF ($p = 2$)	0.848	0.834	0.814	0.803	0.78
	MF ($p = \infty$)	0.806	0.817	0.792	0.786	0.780
CROSS-BIN	KS	0.859	0.854	0.837	0.814	0.817
	MATCH	0.865	**0.882**	0.865	0.851	0.837
	EMD	0.772	0.749	0.732	0.738	0.721
	QF	0.862	0.856	0.845	0.823	0.823

for their comparison bin-by-bin measures are used, too. From an algorithmic point of view these measures can be considered as rigid, allowing no error-tolerance. A small degree of error-tolerance is obtained by considering bins that reduce noise by grouping distances, a procedure which however is coming with the discontinuity on bin-boundaries problem. Cross-bin measures instead allow a comparison of bins having different index, hence allowing for a much higher degree of flexibility and error-tolerance. Regarding the results in Table 7.7, it turns out that representing a protein binding site by 28 instead of two histograms leads to a win of accuracy by holding the complexity unchanged. The win of accuracy can be explained by a better representation of the geometry of protein binding sites. In fact, still a consideration of distances between pseudocenters is performed, however, due to the connection of distances and physicochemical properties the shape of a protein can be captured in a much preciser way since the triple *property-distance-property* leads to a considerably finer model of the protein binding site. In terms of the used type of measure there are only slight differences with a tendency towards cross-bin measures. At a first sight this is astonishing since bin-by-bin measures allow almost no error-tolerance (except this introduced by binning), whereas cross-bin measures are rather flexible and have a higher ability to handle noise, struc-

tural differences and mutations. Hence, one would expect a significantly better performance of cross-bin measures that, however, is not recognizable in the results obtained; an effect maybe caused by the discontinuity on bin-boundaries problem which leads to an overall bad performance.

Therefore, in the next experiment still the representation based on 28 histograms is used, however, instead of using bins and counts to generate the histograms, fuzzy histograms are used, hence counts are substituted by sigma-counts. The results obtained by applying this representation (for $\sigma = 1$ Å) and different measures on histograms are summarized in Table 7.8. Here another

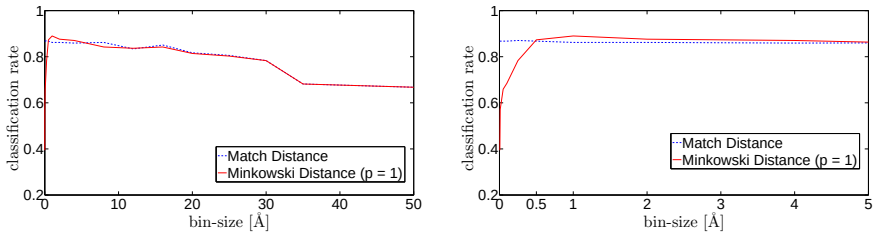
Table 7.8: Overview on the results obtained by using fuzzy histograms and different measures on these histograms.

	k	1	3	5	7	9
BIN-BY-BIN	JACCARD	0.885	0.890	0.862	0.851	0.842
	MF ($p = 1$)	0.890	**0.901**	0.879	0.862	0.845
	MF ($p = 2$)	0.890	0.876	0.870	0.845	0.839
	MF ($p = \infty$)	0.868	0.851	0.814	0.811	0.797
CROSS-BIN	KS	0.859	0.839	0.834	0.811	0.792
	MATCH	0.862	**0.879**	0.856	0.845	0.834
	EMD	0.839	0.839	0.834	0.811	0.792
	QF	0.853	0.851	0.848	0.828	0.825

result is obtained: Bin-by-bin measures perform clearly better than cross-bin measures. Hence, a stronger error-tolerance is becoming even counterproductive in this case. Where in the crisp case cross-bin measures make use of their ability to process bin-boundaries, in the fuzzy case such boundaries do not appear so that cross-bin measures result only in a higher error-tolerance, which obviously becomes too high. However, in comparison to the non-fuzzy approach, classification rates clearly could be increased.

In another experiment, the chosen bin-size is evaluated to demonstrate that a bin-size of 1 Å is not only a reasonable choice, but does also lead to acceptable results. To demonstrate this, first the crisp histogram representation is used, where properties and distances are considered jointly. As measure of choice the Jaccard coefficient and the match distance are chosen, respectively, as representatives for bin-by-bin and cross-bin measures. The classification

rates on the ATP / NADH dataset are recorded for different bin-sizes in the interval $[0.005, 50]$ Å and illustrated in Figure 7.9. They indicate that for the purpose of classifying structures from this set, a bin-size of 1 Å is not necessarily the optimal choice. However, as already mentioned, a search for the



(a) Classification rates vs. bin-sizes in the interval 0.005 to 50 Å

(b) Classification rates vs. bin-sizes in the interval 0.005 to 5 Å

Figure 7.9: Classification rates of the histogram approach which considers the properties and distances jointly vs. used bin-sizes; results for the 1-NN classifier and a bin-by-bin and a cross-bin measure, respectively.

best combination is very time consuming, moreover, it leads to an overfitting which is undesired. Instead, a choice of 1 Å seems to be reasonable, moreover, this choice is located within the interval $[0.5, 5]$ Å leading to very good results. From an algorithmic point of view, the obtained results are very interesting, too. Bin-by-bin measures perform very poor if small bin-sizes are chosen, however, increase their performance quickly. With an increasing bin-size the classification rates fall slightly until a critical size is reached, where the classification rate falls strongly. The bad performance obtained by using small bin-sizes can be explained easily: The bin-size can be considered as a degree of error-tolerance. Choosing small-bin-sizes, a small degree of error-tolerance is allowed, since even a small variation of an observed distance can lead to an assignment to a different bin. In the extreme case (bin-size $\rightarrow$ 0 Å), each distance is assigned to its own bin, thus, almost all comparisons have maximal distance in the case of noisy data that is moreover subject to structural flexibility. In this regard, cross-bin measures and in particular measures based on cumulative distributions are more robust, since they do not require that bins match exactly. Here the small degree of error-tolerance realized by a small bin-size is abrogated by the usage of a cross-bin measure. From a certain bin-size on, as approx. 35 Å in the case considered here, the reached error-tolerance is becoming so high that almost all protein binding sites are considered as equal

leading to the bad classification rates.

Fuzzy histograms allow for the adjustment of two parameters, namely the bin-size and the width of the fuzzy sets σ, latter that was originally set to 1 Å. Both parameters are evaluated in terms of a grid and the results are depicted in Figure 7.10. Here a similar result is obtained as in the case of crisp histograms.

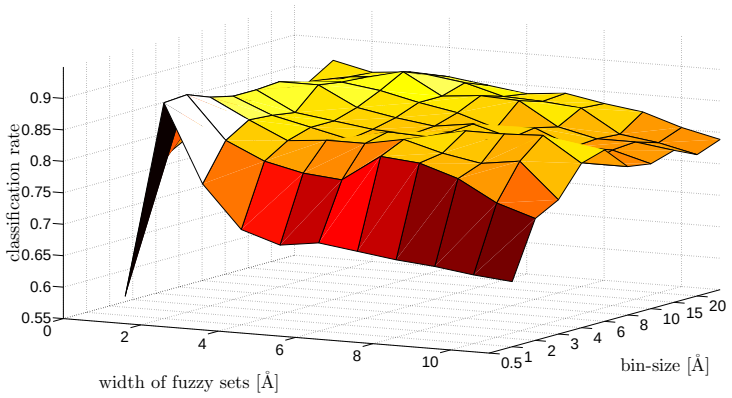


Figure 7.10: Classification rates obtained by using the fuzzy histogram approach and the Minkowski ($p = 1$) measure in dependence of the bin-size and the width of the fuzzy sets.

Bad classification rates are obtained for bin-sizes below 1 Å. A "peak" is obtained for bin-sizes in the interval $[1, 2]$ Å from which on the classification rates slightly fall with increasing bin-size. The classification rates do also decrease slightly with increasing width of the fuzzy sets. This is due to an increased error-tolerance which is becoming from a certain degree on counterproductive since even dissimilar structures become similar. In the other case, if very small widths are used the classification rate also falls, now to the level of the non-fuzzy approach. Cross-bin measures lead to similar results with the difference that very small bin-sizes do not lead to very low accuracies.

The results obtained on histograms can be summarized as follows: The variant in which physicochemical properties and distances are considered separately performs astonishing well, however, since its complexity is equivalent to the complexity of the approach in which both information are considered jointly and because the accuracies of the latter approach are much higher, the former approach is discarded in favor of the latter one. Regarding distance measures, it sticks out that even though of high complexity, the quadratic form and the earth movers distance are not superior to more efficient measures

like the match or the Kolmogorov-Smirnov distance. In the case of bin-by-bin measures one can summarize that all measures perform well, except the Minkowski form distance with parameter $p = \infty$. The problem appearing, if this setting is used was already discussed. It results from the focus on that bin which maximizes the distance and the ignorance of all other bins. Hence, the parameter setting which is not subject to this phenomenon, i.e. an up-weighting of larger distances, is the Minkowski form distance with parameter $p = 1$. Therefore, the cross-bin measures *match* and *Kolmogorov-Smirnov* distance and the bin-by-bin measures *Minkowski form* ($p = 1$) and *Jaccard* will be used in the following of the experimental study.

Feature Vectors

Feature vectors were introduced as an extension of histograms. In particular, simplices were proposed due to their ability to represent the surface of a 3-dimensional curved object. Again, like in the case of histograms the parameter giving the bin-size is first taken out of consideration and instead a reasonable choice of bin-size 1 Å is taken. For feature vectors only a small set of measures was proposed, since entries of these vectors are not endowed with a metric structure so that the usage of cross-bin measures does not make sense. Hence as measures of choice the Jaccard coefficient and the Hamming similarity are used[4] for which the results obtained are summarized in Table 7.9. However, the results are disappointing: Even though the polynomial complexity of the feature vector approach is increased by order one in comparison to the histogram approach, the results are not significantly better, in fact they are slightly worse. Regarding the chosen measures on vectors it clearly sticks out that the Jaccard coefficient is the better choice for comparing these feature vectors, mainly due to the following reason: Where the Hamming similarity is also

Table 7.9: Classification rates on the ATP/NADH dataset obtained by using feature vectors and different measures on vectors.

k	1	3	5	7	9
HAMMING	0.834	0.811	0.789	0.749	0.730
JACCARD	**0.868**	0.851	0.823	0.817	0.803

[4]There exist a vast number of similarity measures which, however, can be applied on vectors. A comprehensive summary is given in (Deza and Deza, 2009).

rewarding common zeros, hence the absence of a certain simplex, the Jaccard coefficient is rewarding only the common presence of a simplex what clearly seems to be the better principle to measure similarity, especially because the set of simplices is very large. The removal of simplices which cannot appear in the 3-dimensional space does not help in this case, since it does not influence the Jaccard coefficient and leads to a linear shift of the similarities returned by the Hamming measure. In fact, the removal of unobservable simplices only leads to a significantly win of efficiency. However, in the case of histograms a win of effectiveness could be obtained by considering fuzzy bins instead of crisp bins. For simplices the discontinuity on bin-boundaries problem is even enhanced since each simplex is coming with three bins, each of which that is subject to this problem. Therefore, in the next step fuzzy feature vectors are considered (with fuzzy width $\sigma = 1$ Å) for which only one of the measures considered in this thesis turns out to be appropriate, namely the Jaccard coefficient. The results obtained by using this measure and this kind of vectors are given in Table 7.10, where a slight loss of accuracy can be observed. Hence, compared to the

Table 7.10: Classification results on the ATP/NADH dataset obtained by using fuzzy feature vectors.

k	1	3	5	7	9
JACCARD	**0.823**	0.780	0.744	0.744	0.800

simple fuzzy histogram approach this extended feature vector representation does not lead to a win of accuracy by requiring even a higher runtime.

This result is hard to explain, especially because simplices should be the most appropriate features to describe a curved surface. An obvious disadvantage is that the set of features is very high, in the case of edge weights up to 11 Å, bins of size 1 Å and 7 physicochemical properties, as considered here, a set of $65,513$ features is obtained. During calculation of the feature vector a single shift of one pseudocenter in the protein binding site leads to the assignment of a large number of simplices to different positions in the feature vector. Therefore, the feature vector approach is much more sensitive to structural flexibility and mutation than the histogram approach is, as illustrated in Figure 7.11. This high sensitivity is confirmed by Figure 7.12 where indeed the majority of obtained similarities is distributed in the interval $[0.08, 0.25]$ in the case of the Hamming similarity and $[0.02, 0.1]$ in the case of the Jaccard co-

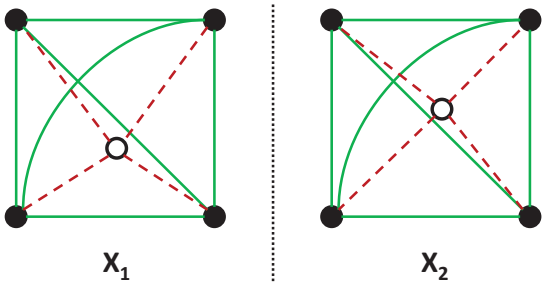
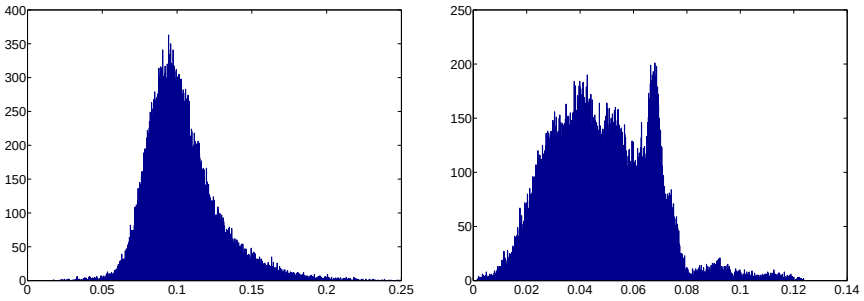


Figure 7.11: Influence of noise: The position of the non-filled pseudocenter is different in the structures X_1 and X_2, therefore, some features do not match. In the case of histograms, 6 of 10 distances match (green lines) leading to a similarity of 6/10. In the case of feature vectors there can be drawn $\binom{5}{3} = 10$ simplices from the structures, however, only such simplices match which do not contain a red dashed edge. The number of these simplices is 4, hence, a similarity of 4/10 is obtained.

efficient. Compared to other measures, the similarities are surprisingly low. This even indicates that the set of common features is clearly dominated by the features which are not common. The behavior that all pairs of structures are considered as dissimilar (expect those pairs that are identical) is obviously undesired in the case of a dataset containing structures sharing a certain degree of similarity. However, some differences between Hamming and Jaccard measure



(a) Hamming similarity (b) Jaccard coefficient

Figure 7.12: Distribution of similarities generated by performing an all-vs-all comparison on the ATP/NADH dataset using crisp feature vectors and, respectively, the Hamming or the Jaccard measure. The peak in position 1 resulting from the comparison of identical structures is not shown.

can be recognized: The scores the Hamming similarity returns are larger than the scores returned by the Jaccard coefficient, an effect obviously caused by rewarding common zeros. Yet, the histogram approximating the distribution

of the Jaccard scores seems to be a result of joining two distributions, namely the distribution of scores obtained by comparing pairs of protein binding sites from different classes and another distribution of scores obtained when comparing binding sites of equal class. The histogram summarizing the Hamming scores must contain both distributions, too, but they intersect completely. This obviously leads to the effect that Hamming performs worse than Jaccard in the classification experiment.

In the next experiment, the bin-size and cut-off distance are considered to evaluate if the choice of 1 Å for the bin-size and 11 Å for the cut-off distance δ are justified or if the bad accuracies result from a wrong setting of the feature vector approach. The bin-sizes are considered in the interval $[0.1, 20]$ Å, where 0.1 Å is a technical lower bound since values below 0.1 Å lead to a very large set of features, even not addressable by a 32-bit integer. On the other hand, bin-sizes of above 20 Å do not lead to high classification rates, since unequal structures become equal in this case, as already shown for the the case of histograms. The cut-off distance δ is evaluated in the interval $[1, 70]$ Å, where again values of above 70 Å would lead to a very large vector, hence, to unacceptable runtimes. To evaluate the different settings, a grid is used and the classification rates are recorded for each grid point, resulting in the plot given in Figure 7.13. Obviously, the parameters are important for the accurate behav-

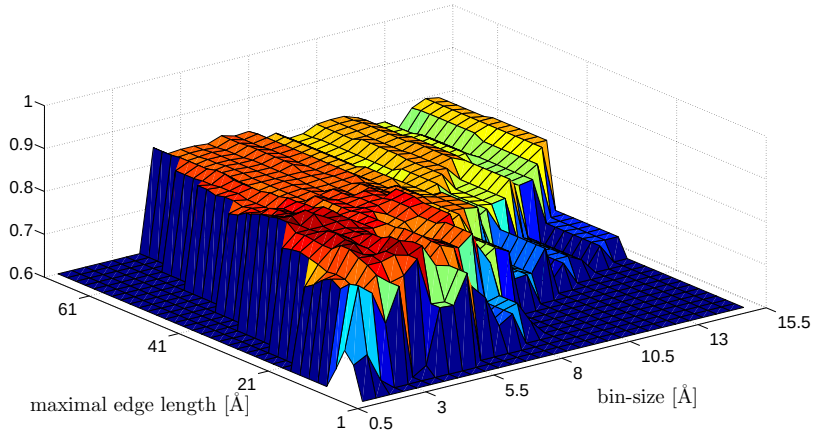


Figure 7.13: Classification rates of the feature vector approach vs. bin-sizes and maximal edge length δ; results of 1-NN and the Jaccard measure.

ior of the similarity measure. However, it turns also out, that the classification rates are stable within a certain range, in particular, that the used setting of

bin-size 1 Å and cut-off value 11 Å is inside this range. Choosing too small bin-sizes combined with larger δ values leads to very low classification rates. In fact rates of around 0.6 have technical reasons, since the similarity matrices consist of zeros due to a failure of the similarity measure. Choosing bin-sizes larger δ, eventually all vectors become very similar, an effect leading to similarity matrices consisting almost of ones. In between these extremes it turns once again out that the classification rates are reduced successively with increasing bin-size.

Summary

Approaches based on feature vectors seem to be a good alternative to more complex methods. These methods lead to acceptable classification rates of above 90% on a non-trivial classification problem, which clearly indicates an effective measure. The assumption that simplices are better features to represent a curved surface is not supported by the results. However, this does not necessarily indicate that simplices are bad features. In fact a central problem of simplices is that there exist an enormous number of them. Hence, even for two similar protein binding sites the probability for observing a certain simplex in one of the binding sites that is not contained in the other one is quite high. This effect eventually leads to skewed and overall low similarity values between protein binding sites. Histograms, on the other hand, consider only a small set of features. Even though these features are very coarse the reduced number is clearly an advantage.

7.3.4 Investigation of Graph-based Approaches

Finally graph-based approaches are considered in this section. Graphs are characterized by their high degree of flexibility which can become a strong advantage since the handling of mutations, structural deformations and noise becomes possible. To enable this, however, a very high number of degrees of freedom must be considered which can make calculations more complex. Here different realization of graph-based algorithms are considered, namely approaches based on the R-convolution framework, subgraph isomorphism and the graph edit distance. The graphs are constructed as described in Section 2.1.2, where, if not mentioned differently, δ is set to 11 Å as recommended by Weskamp et al. (2007).

Measures based on the R-Convolution Framework

In this thesis three measures were discussed that follow the R-convolution framework, namely the RW-, SP- and extended SP-kernel. Before presenting the results obtained, first a distinction between kernel and metric is discussed. Kernels resulting from the R-convolution framework come with a central problem: The calculation is realized as an all-vs-all comparison, as visualized in Figure 6.1, which obviously leads to an averaging effect. Moreover, due to the additive aggregation of the values obtained by comparing the substructures, the resulting scores are growing with size of the graphs to compare. Such a dependence between score and size of the structures is generally not desired. Therefore often normalization techniques are applied. E.g., if the number of substructures n_s and the maximal score ms a comparison between two substructures can return is known, a normalization can be performed easily by dividing the kernel score obtained by the product of n_s and ms. However, in this case a contrary result is obtained due to a large number of comparison between dissimilar substructures. Here the score decreases now with increasing size of the graphs to compare, even in the case of similar graphs. This is still a behavior that is undesired for a measure on protein binding sites. Many authors therefore propose to use a mapping which is transforming the kernel into a metric and which is reducing the averaging effect. To transform a kernel into a metric, Hoffmann et al. (2010) proposed to use the mapping

$$d(x,x') = \sqrt{k(x,x) + k(x',x') - 2 \cdot k(x,x')}\,, \tag{7.2}$$

where k is the kernel and d the resulting metric. By considering $k(x,x)$ and $k(x',x')$, hence the similarities to itself, this equation is introducing a normalization, thus, should be a better measure than the pure kernel. In the following experimental study, however, the kernels are considered in their pure form, since preliminary experiments (not shown here) indicated that there are almost no differences in the accuracies obtained by applying, respectively, normalized or raw kernel values. The results obtained are summarized in Table 7.11 indicating that at least for a comparison of protein binding sites methods based on the R-convolution framework are inappropriate. However, an interesting observation can be extracted from the experiments: If the substructures are compared by using a $0/1$ measure as the Dirac-kernel, even in the case of similar structures there will be produced many zeros. This is due to the fact

Table 7.11: Classification results on the ATP/NADH dataset obtained by using different measures based on the R-convolution framework.

k	1	3	5	7	9
RW-KERNEL	0.597	0.597	0.597	0.608	0.608
SP-KERNEL	0.594	0.563	0.586	0.603	0.603
SPSA-KERNEL	0.625	0.617	0.625	**0.775**	0.685

that in the most cases the drawn sequences of kind (node label, edge weight, node label, ...) will not completely match. Hence overall small similarities are obtained caused by summing-up these zeros. This behavior is alleviated by using another measure, namely the measure based on sequence-alignment. Here still many unequal sequences are compared all-vs-all, however, they contribute to the score even if they do not match completely. In particular, in the case of similar binding sites one can expect that the sequences contain at least equal subsequences which will contribute to the score eventually leading to a high overall score.

Measures based on the Maximum Common Subgraph

The standard concept to measure equivalence between graphs is graph isomorphism that however is a binary measure not suitable for the comparison of graphs representing protein binding sites. A relaxation which was discussed in this thesis is subgraph isomorphism and the maximum common subgraph (mcs) which can be obviously used as a similarity measure and can be calculated by means of the Bron-Kerbosch (BK) algorithm or the local clique (LC) heuristic. However, subgraph isomorphism is still a concept that is rigid allowing only for a small degree of error-tolerance. Therefore, instead of searching for the mcs, a search for the maximum approximate common subgraph (macs) was proposed realized by means of the local clique merging (LCM) heuristic. Using this concept, a win of flexibility is obtained whose strength is adjustable by the parameter γ[5]. Here values of 0.9, 0.8, 0.7 and 0.6 are considered, where 0.6 leads to a very flexible measure allowing a large degree of structural flexibility. Since the setting $\gamma = 1.0$ leads to the classical mcs measure, calculations are performed by means of the BK algorithm in this case. Moreover, the local

[5]In fact, this approach requires the specification of three parameters, namely γ, γ' and ω. In this experimental section γ' is set to $\gamma - 0.2$ and ω to 0.4. Hence, only the parameter γ is considered.

clique approach is evaluated here as an efficient alternative to the BK algorithm which, however, will not guarantee any quality of the solution since it is purely heuristic. The parameter δ specifying the maximal edge weight in the graphs is set to infinity, thus, the graphs representing protein binding sites are complete. This is also the model often used in mcs related literature.

The mcs as well as the macs can be easily embedded into the OWA framework to increase flexibility even more (cf. Section 2.4.1). Therefore, in the experimental study in sum three parameters are considered, namely k of the classifier, γ used to relax the concept of mcs and finally λ required in the OWA-framework. The results on the classification experiments are summarized in the Tables A.1 up to A.6 in the appendix of this thesis. In Table 7.12 the results are given in a more compact way for $\lambda = 1.0$, which is the optimal λ-setting. Even though, the OWA-framework could increase accuracies in the case of la-

Table 7.12: Classification rates on the ATP/NADH set using common subgraph approaches for different values of k and $\lambda = 1.0$

k	1	3	5	7	9
$\text{LCM}_{\gamma=0.6}$	0.682	0.628	0.623	0.606	0.597
$\text{LCM}_{\gamma=0.7}$	0.862	0.837	0.837	0.814	0.808
$\text{LCM}_{\gamma=0.8}$	0.839	0.825	0.811	0.808	0.797
$\text{LCM}_{\gamma=0.9}$	0.851	0.825	0.820	0.811	0.794
BK	**0.865**	0.851	0.839	0.831	0.831
LC	0.848	0.820	0.817	0.800	0.780

beled point clouds, it fails in the case of the graph-based approaches as can be taken from the tables in the appendix. Where in the case of labeled point clouds each inclusion used in the OWA-framework was optimized separately, in the case of m(a)cs this framework does only affect the normalization: In the case of $\lambda = 0.0$ the size of the m(a)cs is divided by the size of the smaller graph, whereas in the other extreme of $\lambda = 1.0$ a division by the larger graph is performed, which obviously performs best. Even though this is not fully in accordance with the expectation (i.e. a match of a substructure containing the most important catalytic residues should be also assigned to a high score), an explanation of this result is possible:

It is quite unlikely that the LigSite algorithm performs so bad that large

fragments of a protein are considered by mistake as part of its binding site. On the other hand, the implication of equal function does surely not hold, if a small binding site is contained in a larger one. On the contrary, compared to the smaller binding site, the additional amino acids surrounding the larger one will lead to a different binding-behavior.

Considering the relaxation introduced by the γ parameter, the results on the classification experiment clearly indicate that a certain degree of flexibility can increase the classification rates (as $\gamma = 0.7$ in the experiment considered). Beyond a certain degree of error-tolerance the classification rate drops strongly. Here, the error-tolerance of the similarity measure becomes too high, so that even dissimilar graphs are considered as similar. However, by using the BK algorithm which is detecting the maximum clique the highest accuracies are obtained. There are two possible reasons for the improved results: Either the rigid concept of the mcs is the better principle to measure similarity between protein binding sites or the worse accuracies result from the application of an heuristic which does not guarantee any quality. To evaluate this, Table 7.12 also gives the results of the LC approach, here used as an heuristic for the detection of the maximum clique. Obviously, this heuristic performs slightly worse than the original approach, in which the BK algorithm is employed, an indirect prove that the LC heuristic is not able to find the maximum clique reliable. This also indicates that LC will probably not be able to find all maximal cliques reliably, hence, that the merging is not performed on optimal start-solutions and thus, that the found quasi-cliques are also not necessarily the optimal solutions.

To further evaluate the efficiency and reliability of the LC algorithm, an additional experiment is applied. In this experiment randomly 200 pairs of graphs from CavBase are selected which are used to calculate 200 product graphs. On these product graphs the maximum clique is detected, respectively, by applying the original implementation of the BK algorithm and as alternative the heuristic based on the detection of local cliques. By dividing the size of the largest clique found by the heuristic by the size of the maximum clique detected by the BK algorithm, for each instance a value is obtained, indicating the quality of the LC approach on that instance. In case of a one, the heuristic performs equal to the exact algorithm, the smaller the value, the more inexact the heuristic becomes. Figure 7.14 summarizes the 200 results obtained in terms of a boxplot. Obviously, the LC algorithm performs quite well. Almost all numbers in the data are one, there exist only some outliers. Given this be-

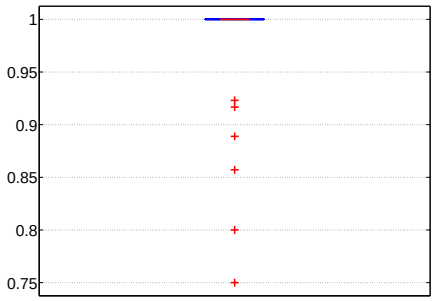
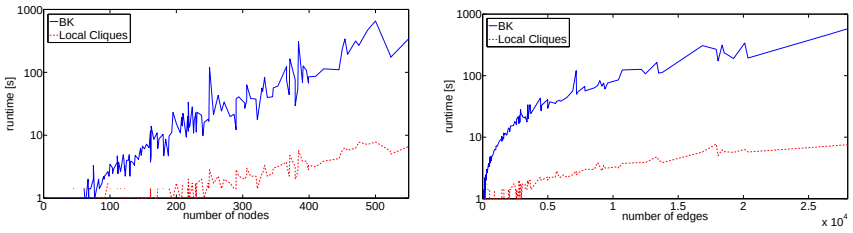


Figure 7.14: Relative deterioration on a set of 200 randomly chosen pairs of protein binding sites.

havior it is very astonishing that such slight differences can have such a strong influence on the classification result. Regarding runtimes which are illustrated in Figure 7.15 the expected result is obtained: LC performs much more efficient than the BK algorithm. Moreover it turns out that the runtime does stronger



(a) runtime vs. number of node (b) runtime vs. number of edge

Figure 7.15: Runtime needed to detect cliques by using the BK algorithm and the local clique approach.

depend on the number of edges than on the number of nodes. This becomes clear by considering the algorithms: The neighborhood graphs used to calculate the local cliques are the larger the more dense the graph is, the search-tree constructed by the BK algorithm has the more branches in this case.

Finally, the CavBase approach is considered which applies the BK algorithm as a subprocedure and which is using the so calculated cliques as seed solution for the calculation of the final score. The results obtained are summarized in Table 7.13. Compared with the former results, CavBase is not able to increase the classification rates, even though it is using the cliques (mcs) and tries to improve them by applying further quite expensive steps. In particu-

Table 7.13: Classification rates on the ATP/NADH dataset obtained by using CavBase scores for different values of k.

k	1	3	5	7	9
CAVBASE	0.817	**0.831**	**0.831**	0.811	0.794

lar it is using the cliques to superimpose the surface points of the two protein binding sites. Eventually, this superposition is used to derive the final score. The result obtained is astonishing, since performing additional steps and using additional information leading to a much finer representation of the protein binding site does not pay off. However, CavBase considers inclusion instead of equivalence to derive the final score. It was already mentioned that a strict equivalence performs better than inclusion in the case of a graph-based representation. Therefore the CavBase results (Table 7.13) are also compared to the the pure BK algorithm (Table 7.14), where in the latter case the size of the smaller graph is used for the purpose of normalization. Hence, inclusion is

Table 7.14: Classification rates on the ATP/NADH dataset obtained by using the mcs approach with setting $\lambda = 0.0$.

k	1	3	5	7	9
BK	0.817	0.823	0.825	**0.828**	0.820

considered, specified by setting λ to 0.0. Here indeed CavBase is able to increase accuracy due to the usage of additional information. This again confirms the assumption that the additional amino acids which are excluded by applying a measure based on inclusion can cause a different binding-behavior of the binding site.

Figure 7.16 is summing up the results of the novel graph-based methods for $k = 1$ and different values of λ, where CavBase is used as baseline. Indeed, all concepts perform the better the closer λ is set to 1.0, which is specifying strict equivalence. Using this setting all novel methods are able to reach classification rates higher than the accuracy of CavBase and higher than majority voting which would lead to a classification rate of 60.28%. The only exception is the LCM approach with setting $\gamma = 0.6$ which obviously is too flexible a setting for a similarity measure.

Finally, to enable some insights into the measures and an interpretability of

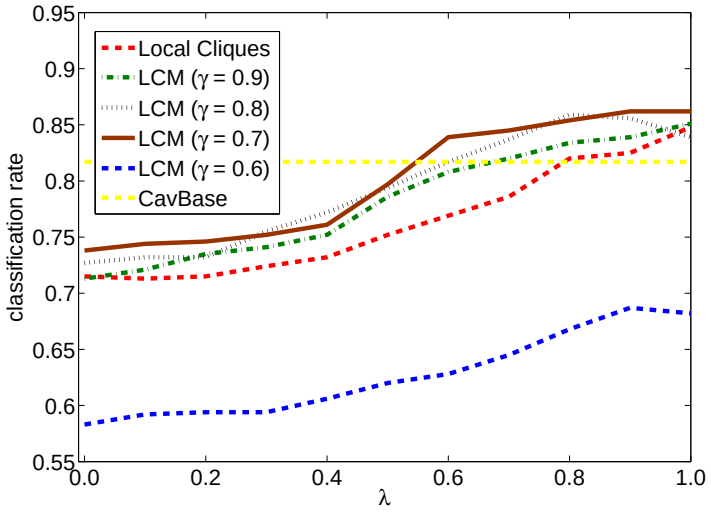
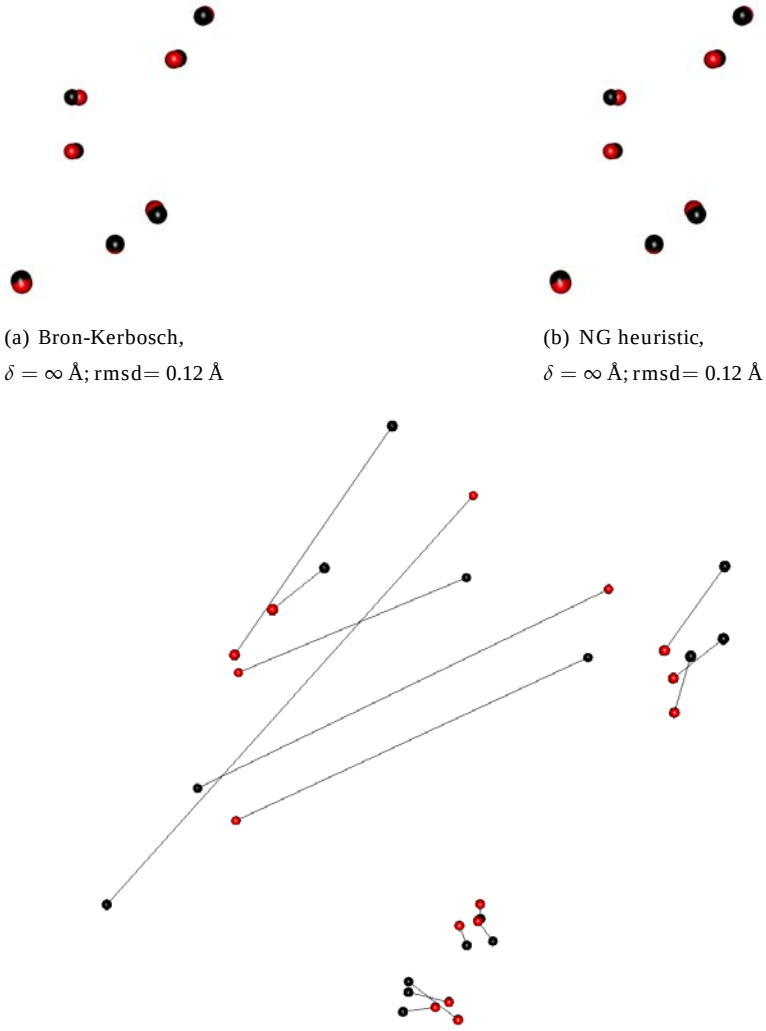


Figure 7.16: Classification rates of different graph-based measures vs. λ.

their behavior the constructed one-to-one correspondences are considered after superimposing them by using the Kabsch algorithm. Here again the same structures are used as in the case of the geometric approach. As one can easily recognize in Figure 7.17, the maximum common subgraphs found by the BK algorithm and the heuristic based on local clique detection are identical, independent of the used algorithm (at least on the chosen instance). Even though the common subgraphs are quite small, they match almost perfectly (given the ϵ-constraint). In Section 2.1.2 an alternative model was introduced representing protein binding sites by incomplete graphs instead of complete ones, realized by setting $\delta < \infty$ Å. This approach is also evaluated in Figure 7.17, where $\delta = 11$ Å is used. Here it sticks out that the size of the detected cliques increase, however, at the cost of a strongly increased rmsd value. This value clearly indicates a bad one-to-one correspondence between the nodes resulting from a strong deformation of the structures needed to enable a match. Technically this phenomenon is realized by removing edges. Since non-existing edges are considered as match during calculation of the product graph, this graph is becoming more dense. Hence, it is likely that there are larger cliques contained in the product graph. However, edges in the product graph represents *equal geometry*, thus geometry is not taken into consideration for distances of above δ Å. This results in the fact that the structure of the protein binding site can become skewed to reach the maximal possible match, an effect which can be



(a) Bron-Kerbosch, $\delta = \infty$ Å; rmsd= 0.12 Å

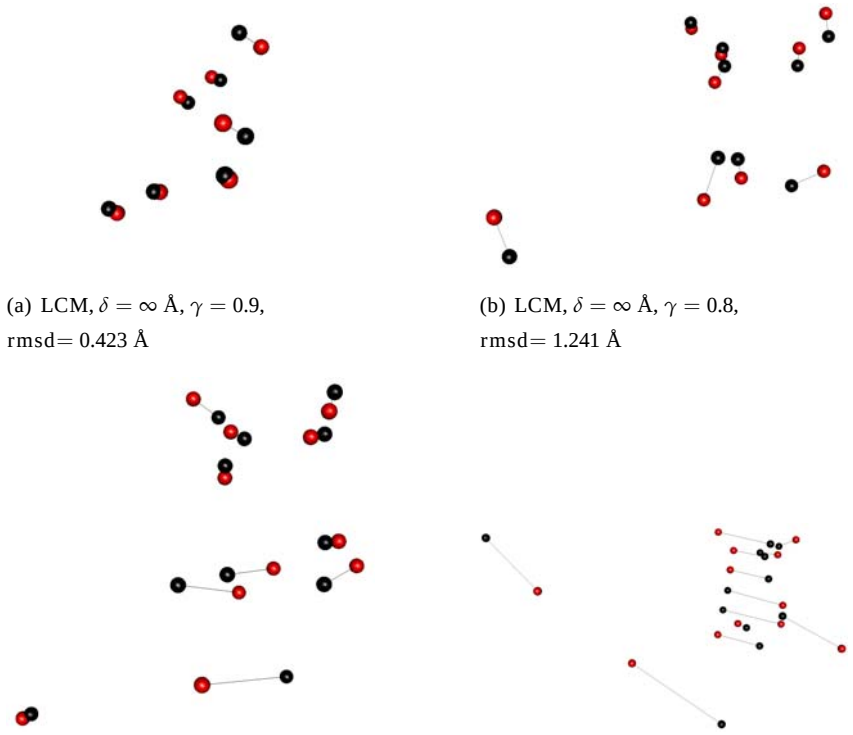
(b) NG heuristic, $\delta = \infty$ Å; rmsd= 0.12 Å

(c) NG heuristic, $\delta = 11$ Å; rmsd= 10.42 Å

Figure 7.17: Visualization of the maximum common subgraph detected in the graphs representing the binding sites `1dy3.1` and `1j1z.10`. The colored balls indicate the pseudocenters of `1dy3.1` (red) or `1j1z.10` (black), respectively. Straight lines are used to indicate the one-to-one correspondences.

recognized especially in Figure 7.17 (c) where pseudocenters located in completely different positions are mapped onto each other. Moreover, due to the increased number of edges the runtime is also increasing. In case of the BK algorithm even a memory overflow appeared, so that calculations could not be performed. A better way to realize a higher degree of error-tolerance is the approach calculating quasi-cliques, hence maximum approximate common sub-

graphs. The results of this approach are visualized in Figure 7.18. Using this



(a) LCM, $\delta = \infty$ Å, $\gamma = 0.9$, rmsd= 0.423 Å

(b) LCM, $\delta = \infty$ Å, $\gamma = 0.8$, rmsd= 1.241 Å

(c) LCM, $\delta = \infty$ Å, $\gamma = 0.7$, rmsd= 2.517 Å

(d) LCM, $\delta = \infty$ Å, $\gamma = 0.6$, rmsd= 3.122 Å

Figure 7.18: Visualization of the maximum approximate common subgraph detected in the graphs representing the binding sites `1dy3.1` and `1j1z.10`. The colored balls indicate the pseudocenters of `1dy3.1` (red) or `1j1z.10` (black), respectively. Straight lines are used to indicate the one-to-one correspondences.

approach it is indeed possible to vary the degree of flexibility gradually by the parameter γ. This allows for the detection of common subgraphs which match almost perfectly, but also for the detection of subgraphs which match only if one of the subgraphs undergoes a certain transformation. This transformation however is soft, i.e., there are only local transformations needed. Larger transformations, as e.g. the movement of a node into a completely different region, are omitted. This is a much more natural way to relax the maximum common subgraph measure. By using the other graph model in which edges are removed that exhibit an edge weight above $\delta < \infty$ Å together with the quasi-clique approach leads again to the effect that the nodes are shifted strongly and that nodes that are located closely (in the geometric data) are moved in com-

pletely different directions. This obviously destroys the complete geometry of the considered structure. Another disadvantage using incomplete graphs to model protein binding sites is that the number of edges in the product graph is increasing strongly in this case. This number positively correlates with the runtime, hence, beside becoming more inappropriate such an approach also leads to much higher runtimes.

Measures based on Graph Edit Distance

Algorithms based on the concept of a graph edit distance allow the highest degree of error-tolerance since the underlying idea is a very, not to say the most flexible approach for measuring similarity between graphs. Due to its importance a large set of algorithms for the calculation of the graph edit distance is available. The most appropriate approaches for the task considered here are the greedy heuristic for calculation of graph alignments (IGA) and its pendant based on evolutionary computation (GAVEO). Graph edit distance requires the specification of a set of parameters, i.e. a set of parameters defining the scoring function s given in equation (6.10). As mentioned, the specification of these parameters is very hard, hence, recommendations from (Weskamp et al., 2007) are taken, where the parameters were set as follows:

PARAMETER	ns_m	ns_{mm}	ns_d	es_m	es_{mm}	ϵ
VALUE	1.0	-5.0	-2.5	0.1	-0.2	0.2

This setting was motivated by the fact that the number of edges grows quadratically with the number of nodes. Hence the parameters defining the edge score were down-weighted to ensure that the overall score is not dominated by the edge score. Moreover, the penalty for the insertion of a dummy is smaller than the penalty for changing the node label (or equivalent for accepting a mismatch) since the insertion of a dummy can preserve the structure of graphs whereas a forced shift of a node (to establish the one-to-one correspondence) can destroy it.

Using this set of parameters, first it is interesting to see how the algorithms IGA and GAVEO used to maximize the same scoring function behave. Obviously, the greedy realization should be much more efficient. Hence, a preliminary experiment is applied to test if it makes sense to substitute the greedy heuristic by an evolutionary algorithm. This test is performed by choosing at random 200 pairs of protein binding sites from the ATP/NADH dataset on

which the graph edit distance is calculated, respectively, by using IGA and GAVEO. As a measure for comparison the relative improvement

$$ri(\mathcal{A}_{GAVEO}, \mathcal{A}_{IGA}) = \frac{s(\mathcal{A}_{GAVEO}) - s(\mathcal{A}_{IGA})}{\min(|s(\mathcal{A}_{GAVEO})|, |s(\mathcal{A}_{IGA})|)}$$

is derived, which leads to values below 0 if IGA performs better and to values larger 0 in the other case; e.g., a relative improvement of 1 would mean an increase in score by a factor of 2. The result is summarized in Figure 7.19 where a boxplot is used for this purpose. Even though both algorithms are using the

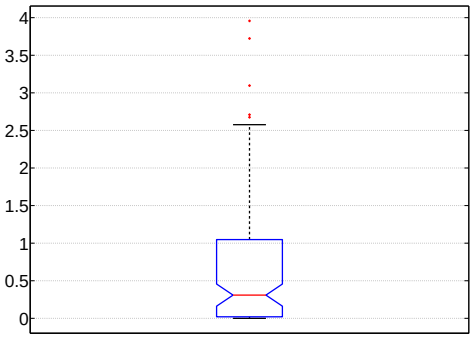


Figure 7.19: Relative improvements obtained on a set of 200 randomly chosen pairs of protein binding sites if IGA is replaced by GAVEO.

same scoring function and -parameters, GAVEO reaches significantly better scores than the IGA approach since the confidence interval ($\alpha = 0.05$) for the mean is above zero. Moreover almost all values are larger zero. Only in one case a negative improvement is obtained which is very small and indicates that the evolutionary search was terminated too early.

However, the higher scores only indicate that GAVEO is the better optimizer, it is still not shown that a better similarity measure is obtained, what would not be the case if, e.g., the increased scores would result from a linear shift. Hence, in the next step it is returned to the classification experiment which allows to assess the quality of the similarity measure in an indirect way. The results on this experiment, given in Table 7.15, allow some interesting conclusions. First of all it sticks out that the results are indeed better if GAVEO scores are used instead of IGA scores[6]. Therefore, GAVEO is not only a better optimizer, it also leads to higher classification rates, hence, should be a better

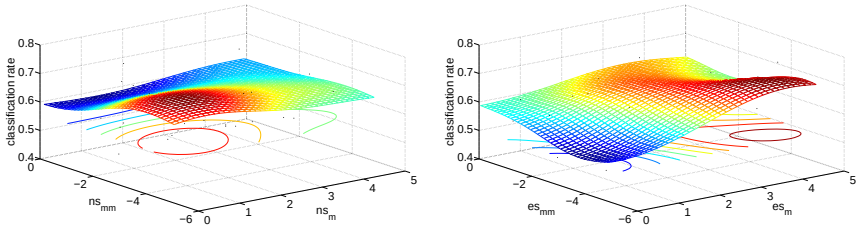
[6]In the case of IGA in some seldom cases a memory overflow appeared, hence a score of $-\infty$ was recorded for the corresponding pair of protein binding sites.

Table 7.15: Classification results on the ATP/NADH dataset obtained by using IGA or GAVEO.

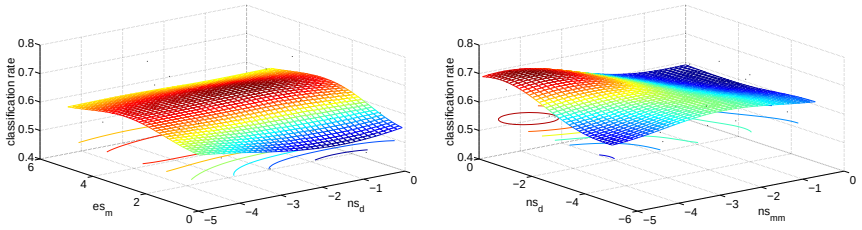
k	1	3	5	7	9
IGA	0.766	0.718	0.724	0.718	0.713
GAVEO	0.789	0.766	0.780	0.786	0.766

similarity measure. A further result is that the high error-tolerance and flexibility of both measures does not increase the classification rate compared to the other measures. A result that is quite interesting, and can be explained maybe by the taken parameterization which sounds in the first moment quite reasonable but which also comes with some problems, as the fact, that in a small graph the number of edges is small, hence that 0.1 (-0.2) becomes too small a parameter for edge (mis)match.

To investigate the chosen parameterization the classification experiment is repeated for the IGA approach where different parameterizations are considered by using the SPOT framework. However, due to an enormous time consumption of this experiment, it is performed on a subset of the ATP and NADH sets, namely on randomly chosen 10%, hence on 14 structures from the ATP and 21 structures from the NADH set. Moreover, to accelerate the calculation, product graphs and cliques are calculated a-priori and reused in each step of the SPOT loop. Since product graph as well as the clique depends on the parameter δ defining the cut-off distance and the parameter ϵ defining the tolerance on edge weights, only the scoring parameters ns_m, ns_{mm}, ns_d, es_m, es_{mm} are considered. For the scoring parameters different intervals are used namely $[0,5]$ for rewards and $[-5,0]$ for penalties. By using latin hypercube sampling a set of 20 parameterizations is generated that gives a good overview about the performance of different parameterizations. Note that SPOT did not performed an optimization, instead it was used to randomly generate a set of parameterizations and to fit models. The thus obtained results depicted in Figures 7.20 and 7.21 are very interesting since they are able to demonstrate the influence of the parameterization of the scoring function on the classification rates. The ROI is chosen in a way which leads in all cases to meaningful parameterizations, nevertheless, the obtained classification results vary strongly, confirming that the appropriate setting of parameterization is very important.



(a) ns_m vs. ns_{mm} (b) es_m vs. es_{mm}



(c) es_m vs. ns_d (d) ns_{mm} vs. ns_d

Figure 7.20: Influence of different settings of the scoring parameters on the classification rate.

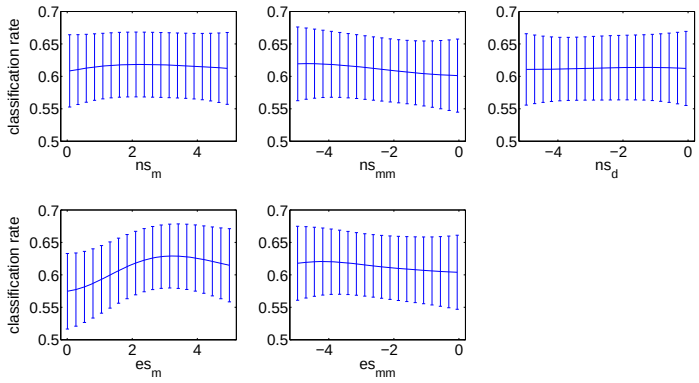


Figure 7.21: Effect plots for each of the five scoring parameters

Choosing the best parameterization out of the 20 tested leads to the parameters given in Table 7.16. They indicate that instead of overweighting the node parameters, the edge parameters should be overweighted, moreover, that a mismatch should be preferred to an insertion of a dummy. Where in the original parameterization a node mismatch was penalized stronger than an insertion of a dummy with the goal to preserve the geometry, the novel parameterization preserves geometry already by overweighting the edge parameters, hence, insertion of dummies can be penalized stronger. However, even

by performing an expensive testing of different scoring parameters, IGA is not able to reach higher classification rates than geometric or feature-based approaches. In Table 7.17 the results are summarized for the case of IGA, where the new parameterization was used. Obviously, the classification rates are for some k slightly higher compared to the original parameterization. However, the strongly increased runtime needed to perform testing of different parameterizations is definitely not justified by the slight improvement. Moreover, it is questionable, if the found parameterization performs well on arbitrary subsets of CavBase since protein binding sites of different volume, number of pseudocenters and distribution of labels must be handled. So it is indeed possible that completely different parameters would perform better in the case of another CavBase subset which contains binding sites with different properties. This would also explain why the gain in accuracy is so low compared to the original parameterization. Moreover, it is very likely that parameters must be adapted onto a certain subset or even onto a certain pair of structures. In fact it would be interesting to consider the scoring function weights as functions of different characteristics of the protein binding sites, instead of using fixed constants. This however is not considered in this thesis, but an interesting starting point for future work.

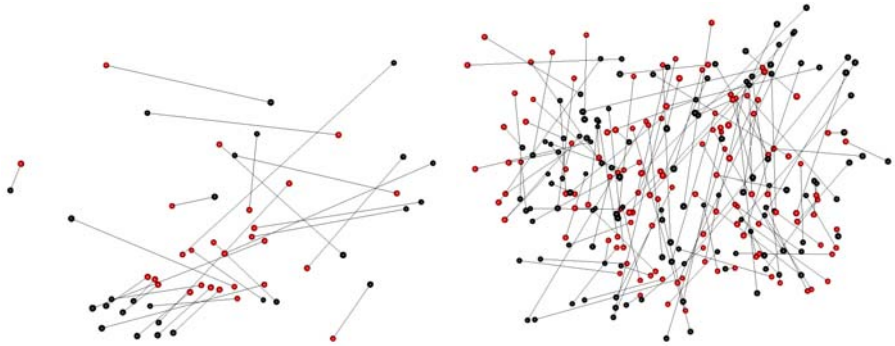
In the final step the generated one-to-one correspondences are considered which are representative for a large set of visually inspected ones. Here again the structures are taken which were already used to evaluate geometric or mc(a)s-based algorithms. After having calculated the alignment the conserved pattern is detected and superimposed by using the Kabsch algorithm. The thus obtained results are illustrated in Figure 7.22. Compared to the conserved structures obtained by applying geometric approaches depicted in Figure 7.8 or methods based on the maximum (approximated) common subgraph in Figures 7.17 and 7.18, a much worse result is obtained. Even though a higher number of pseudocenters can be aligned correctly, the spatial structure is obviously taken more or less out of consideration, which is indicated by the long

Table 7.16: Parameter setting which performs best among the tested parameterizations.

PARAMETER	ns_m	ns_{mm}	ns_d	es_m	es_{mm}
VALUE	1.171	-1.339	-3.360	4.509	-4.255

Table 7.17: Classification results on the ATP/NADH dataset obtained by using iterative graph alignment and the alternative parameterization.

k	1	3	5	7	9
IGA	0.741	0.676	0.721	0.701	0.732



(a) Iterative graph alignment; rmsd= 10.67

(b) Graph alignment via evolutionary optimization; rmsd= 10.92

Figure 7.22: Visualization of the conserved pattern ($\omega = 1, \xi = 1$) obtained by constructing graph alignments of `1dy3.1` (red) and `1j1z.10` (black). One-to-one correspondences are visualized in terms of a straight line.

lines used to illustrate one-to-one correspondences. This is due to the concept of maximizing a scoring function which returns in its original parameterization the higher values the more pseudocenters are correctly mapped onto each other. Even though this can lead to bad one-to-one correspondences between edges, the loss on score is very low due to the absolutely very small parameters used to penalize edge mismatches. However, there are differences identifiable between IGA and GAVEO: Since IGA is using common subgraphs as seed solutions and extends them in a greedy way successively to an alignment, the structure is considered to a much higher degree due to two reasons: First of all it is guaranteed that the mcs is mapped correctly. Moreover, the greedy extension of the partial solution will take in the first iterations such pairs which match perfectly, in physicochemical property and geometry as well. Hence an overall good partial alignment is found which is eventually extended in the last iterations to a complete one by accepting stronger mismatches. GAVEO on the other hand is much more flexible and it is using this property to maximize the score by mapping as much as possible nodes with equal label onto each other

thereby destroying the structure of one protein binding site to enable a match. Insofar the number of equal nodes is counted that is weighted by the parameter ns_m to maximize the overall score. Hence, the calculation of the GAVEO score is somehow similar to the histogram approach in which the physicochemical properties are considered. Indeed, the classification rates are quite similar. Where GAVEO is reaching 78.9%, the approach based on properties-histograms was able to achieve a classification rate of 78.3%. Considering the calculated distances (in the case of histograms) and similarities (in the case of GAVEO) in Figure 7.23, it moreover turnes out that both measures correlate, i.e., the correlation coefficient is -0.9404. However, one should note that GAVEO is still able to detect structurally conserved patterns. In an additional experiment 1000 randomly chosen pairs of protein binding sites were aligned by GAVEO. In addition the mcs of these structures was calculated by the BK algorithm. In an afterwards applied test it was checked if the mcs found by the BK algorithm was mapped correctly in the alignment calculated by means of the GAVEO algorithm. Fortunately, this was the case in 75.5% of the tried pairs. Moreover one should note that a strict test was performed, i.e. that already in the case of one misassignment the test returns a negative result.

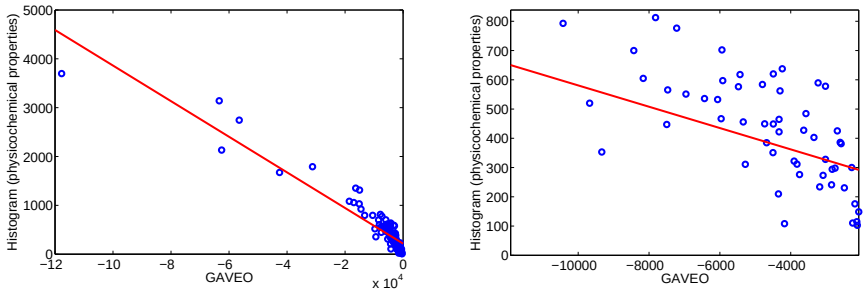


Figure 7.23: GAVEO scores vs. histogram distances; obviously both measures correlate (right figure gives an enlargement of the most populated area).

7.3.5 Summary

In this section different representations of protein binding sites were considered on which different similarity or distance measures were applied. Since it was not possible to evaluate the representations and measures directly, an indirect way was chosen. Here, classification experiments were performed indicating by the reached accuracy the quality of the used representation and the

measure on it. Figure 7.24 summarizes the classification rates of all measures on the ATP / NADH dataset. As a first positive results it turned out that it is in-

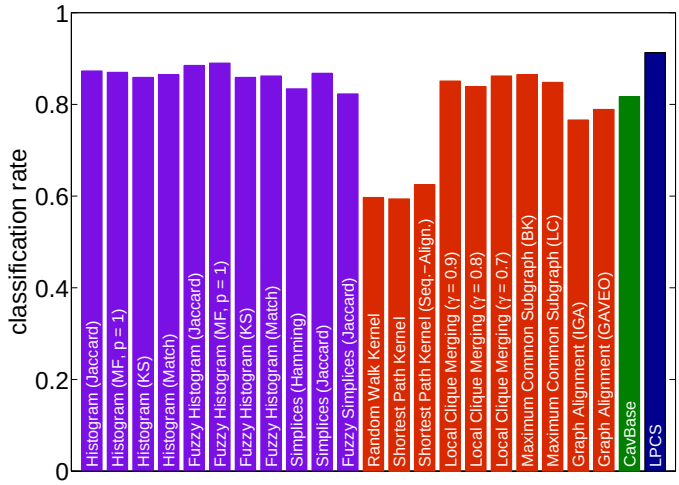


Figure 7.24: Classification results on the ATP/NADH binary classification problem; purple bars represent feature-based approaches, red bars graph-based approaches, finally the blue bar is representing the geometric approach. The result obtained by applying CavBase as similarity measure is given by the green bar.

deed possible to process the CavBase data directly, i.e., without transforming it into another representation. Even though the model used is becoming rigid and the used measure is considering the structures globally, the combination of both was able to reach classification rates of above 90% on a binary classification problem containing 214 “positive” and 141 “negative” instances. Considering the constructed alignments it turned out that even the rigid representation of a protein binding site which is considered globally during calculations does allow to construct meaningful alignments. These alignments have the property that a pattern is contained that matches almost perfectly, indicated by a very small rmsd value. An edit distance derived from this alignment would not change the similar patterns. On the other hand pseudocenters which have no counterpart spatially close would be mapped onto dummys. As a result, this approch would preserve the global structure as much as possible.

A surprising result is that simple counting of equal physicochemical properties leads already to a classification rate of 78.3%. Extending this approach by considering distances and physicochemical properties jointly increases this

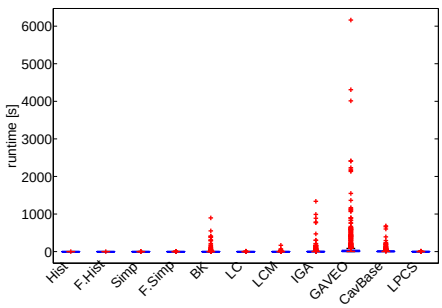
rate to 88.2%, finally by using techniques from the field of fuzzy logic an accuracy of above 90% is obtained. This is astonishing since these histogram-based approaches are rather simple and discarding a lot of information. However, similar approaches can be used which consider considerably more information. Especially simplices were employed in this thesis since they are able to represent pieces of a curved surface. Unfortunately, the experimental result could not confirm the assumption that albeit on the costs of a slightly higher complexity the accurracies can be increased strongly compared with the simpler approach based on histograms. It even turned out, that the accuracies are slightly worse.

Some of the graph-based approaches presented in this thesis perform very poor. Measures based on the R-convolution framework come with a polynomial complexity, however, of highly order. In terms of classification rate they fail completely, a behavior caused by the all-vs-all comparison of substructures. Obviously, the number of substructures is very high since graphs representing protein binding sites exhibit up to 1000 nodes. Performing all-vs-all comparisons will therefore lead to many comparisons of dissimilar substructures, even in the case of two overall similar structures. The concept of graph edit distance is from a theoretical point of view very interesting and indeed it is the most flexible and a very intuitive approach. However, it requires the specification of a set of parameters which have enormous influence on the measure; but finding good parameterizations is not a trivial task, in fact it is even questionable if a good parameterization can be found for a heterogeneous dataset in which graphs of different size appear. Contrariwise, it is likely that parameterization must be considered for each instance. Instead of matching two graphs completely after transforming them appropriately, one can remain the graphs unchanged and try to match them partially, an approach leading to the maximum common subgraph problem which can be solved by means of clique detection. Even though the problem of maximum clique detection is NP hard, heuristics can be also applied to solve it efficiently. Here especially the heuristic which employs local cliques should be emphasized, which has a polynomial complexity. The maximum common subgraph measure is very intuitive and exhibits interesting properties. However, it is quite rigid and does not allow for the search of approximate patterns. By replacing the search for maximum cliques by a detection of maximum quasi-cliques, hence by relaxing the maximum common subgraph measure to the maximum common approxi-

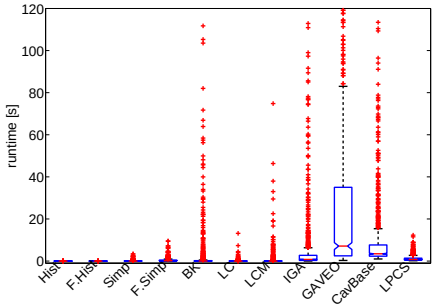
mate subgraph measure, a higher degree of flexibility is enabled which allows for the detection of approximate patterns. Another way to increase flexibility proposed by (Weskamp et al., 2007) and realized by removing edges which exhibit a weight above a certain threshold does not seem to be an appropriate approach since a high amount of structural information is lost. Hence, pseudocenters located in completely different positions are mapped onto each other, since algorithms are not able anymore to recognize that pseudocenters are not located closely. Moreover, since the number of edges in the product graph is increasing in this case, the calculations become more expensive, too.

Overall, the geometric approach performed best in terms of accuracy and rmsd value of the found conserved pattern. However, methods based on the maximum common subgraphs perform in the latter case also well. It is astonishing that feature-based approaches can reach an accuracy comparable to the geometric approach. A possible reason is the usage of a k-NN classifier: Here it is completely sufficient to have some protein binding sites which exhibit the correct class and which are close to that binding site one wants to classify. Therefore, in the next section another experiment is performed, in which a large set of similar protein binding sites is retrieved.

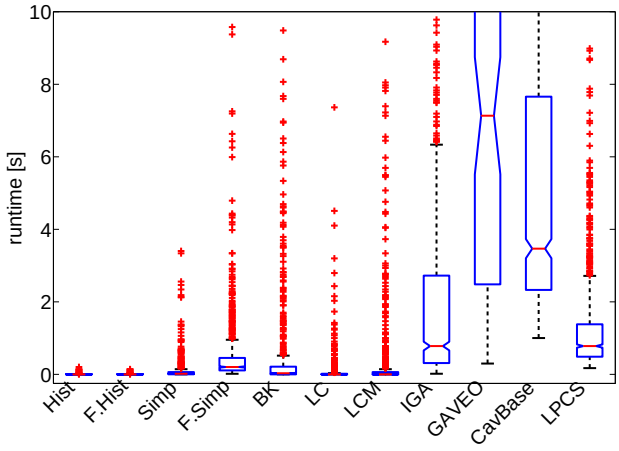
Before introducing the next experiment, first the runtimes are considered. To give an overview on the real runtime 1000 pairs of protein binding sites are selected and the runtime needed for a comparison is recorded for each approach. For calculations a machine equipped with an Intel® Core™ i7 860 CPU (2.80 GHz) and 12 GB memory was used, where for each calculation one core was used exclusively. The distribution of the runtimes collected in this way is given in Figure 7.25, where a boxplot is used for this purpose; Table 7.18 moreover is giving the means and standard deviations. In Figure 7.25 (a) one can recognize that in particular the graph-based algorithms BK, IGA, GAVEO and CavBase lead even to runtimes above 1000 seconds (15 minutes). Such high runtimes are especially critical in the case of larger studies. They are caused by a huge search space if large and medium-size binding sites are processed. All other methods seem to be more robust in this regard. Considering the Figures 7.25 (b) and 7.25 (c) one can moreover recognize that GAVEO, IGA and CavBase are the most inefficient approaches. Even though 50% of the medial runtimes are distributed in the interval $[0.5, 1.6]s$ in the case of LPCS, there are also few outliers up to a runtime of 18 seconds. Among all measures, feature-based approaches perform best in terms of runtime; moreover they scale very



(a) all collected runtimes



(b) details of the distribution in the interval 0–120 seconds



(c) details of the distribution in the practically more relevant interval 0–10 seconds

Figure 7.25: Runtimes of proposed approaches on a set of 1000 randomly chosen pairs of protein binding sites summarized in terms of boxplots. Calculations performed on a machine equipped with an Intel® Core™ i7 860 CPU (2.80 GHz) and 12 GB memory.

well what is indicated by a very low variance.

7.4 Similarity Retrieval

So far, a classification experiment was considered as an indirect assessment of the proposed measures on protein binding sites. However, in CavBase the central task is to detect a set of protein binding sites that are similar to a query protein binding site. Therefore, in the following experiment four protein binding sites are selected, each of which that is used as query which is submitted to

Table 7.18: Mean runtime and standard deviation of different measures on a randomly chosen benchmark set of 1000 protein binding sites. Calculations performed on a machine equipped with an Intel® Core™ i7 860 CPU (2.80 GHz) and 12 GB memory

APPROACH	μ	σ
HISTOGRAM (DISTANCES AND PROPERTIES JOINTLY)	0.0061s	0.0164s
FUZZY HISTOGRAM (JOINTLY)	0.0059s	0.0160s
SIMPLICES	0.0963s	0.2848s
FUZZY SIMPLICES	0.4886s	0.8924s
BRON-KERBOSCH	6.6584s	46.668s
LOCAL CLIQUES	0.0872s	0.5627s
LOCAL CLIQUE MERGING	0.7802s	6.452s
ITERATIVE GRAPH ALIGNMENT	12.295s	74.386s
GAVEO	90.748s	355.12s
CAVBASE	13.605s	45.448s
LABELED POINT CLOUD SUPERPOSITION	1.224s	1.3298s

CavBase. Beside the original CavBase measure, the novel measures proposed in this thesis are used to detect the k most similar protein binding sites. Eventually, for each inquiry and each measure a set of the k most similar protein binding sites is retrieved which is evaluated in terms of the enzyme classification (EC) nomenclature. This nomenclature has been assigned independent of binding site considerations and offers a fair comparison since it is not based on a certain similarity extraction procedure, e.g., a certain measure or representation. Instead the enzyme classification is based only on the enzyme-catalyzed reaction. As particular chemical reactions follow well-defined elementary steps of a chemical transformation the geometry and interaction pattern in the catalytic center of enzymes accelerating the same reaction is highly conserved. Therefore one can expect that similarity in the EC number indicates also some similarity in the binding pocket, at least next to the catalytic center.

The EC nomenclature assigns an EC "number" (also called "class") of the form $\star.\star.\star.\star$ to an enzyme, categorizing the reaction the enzyme is catalyzing. Each '$\star$' of the EC class is a placeholder for an integer. The more right

this integer is placed, the more specific it becomes. Hence, a hierarchical structure is obtained that can be used to evaluate whether the retrieved enzymes are correct hits. Strictly speaking, for each of the *k* retrieved protein binding sites the EC class of its enzyme is looked up. If the protein is not EC annotated (e.g. because it is not an enzyme), it is not considered further and regarded as false hit. If it is annotated, the entire EC number is taken to decide whether a detected hit is reasonable. Even though the EC class is an annotated parameter which is moreover independent of a certain representation or measure one has to keep in mind that such an evaluation is not optimal to decide whether similar geometries have been recognized. As mentioned, the reaction of a protein is accelerated only in a small area (the so-called *catalytic center*), where protein binding sites are indeed highly conserved. This area consists only of few pseudocenters and is responsible, e.g., to cleave substrates at a predefined position. However, the overall binding site geometry of proteins belonging to the same EC number can still vary quite significantly since different substrates of reasonable size are usually recognized. Due to these problems, additional experiments will be performed in this section to demonstrate that a particular enzyme is recognized for structural similarity.

Beside the original CavBase measure all novel approaches will be used in these experiments. However, due to an enormous runtime required for performing this type of comparison, some specific realizations of measure (e.g. certain parameter setting) are not considered further since they turned out to be inappropriate in the previous experiment. For the purpose of a comparison to sequence-based approaches moreover the well-known Smith-Waterman algorithm is used, which is parameterized with the Blosum-62 matrix. For calculations a subset of CavBase is used: Instead of considering all protein binding sites stored in this database, only those binding sites are selected which exhibit a resolution of at most 2.5 Å leading to a set of cardinality 186,507. All distances (or similarities) between the structures in this set and the query are calculated. Having obtained all values, the protein binding sites are sorted in ascending (descending) order according to their distance (similarity) and the top-100 structures are selected. This selection is based on the PDB code taking care that the list only contains unique proteins.

7.4.1 Dataset

With the help of Serghei Glinca from the local department of pharmaceutical chemistry in total four query cavities were selected namely `1eag.1`, `2eu2.1`, `2oq5.1` and `3hec.3`. The query `1eag.1` corresponds to the active site of an aspartic protease (EC `3.4.23.24`), `2eu2.1` to a carbonic anhydrase (EC `4.2.1.1`), `2oq5.1` to a serine protease (EC `3.4.21.-`), finally `3hec.3` represents the active site of a kinase (EC `2.7.11.24`). From an algorithmic point of view these queries are very interesting: It is known that the corresponding enzyme classes can either be rigid or rather flexible with respect to their overall geometric shape. Here an enzyme class will be called rigid if all protein active sites belonging to this class share an almost identical shape; otherwise it is called flexible. Moreover, a distinction between normal/high populated classes and low populated classes is performed. The aspartyl protease EC class `3.4.23.24`, e.g., is low populated since only 11 proteins are contained in the PDB (8 in CavBase) which belong to this class. On the other hand, the class of carbonic anhydrases `4.2.1.1` is normally/highly populated because 511 proteins are contained in the PDB which all share the same EC number. Table 7.19 gives a summary on this properties. Obviously, for a query whose class is rigid a "simple" calculation of the degree of equivalence is sufficient. In the other

ENZYMES	POPULATION	FLEXIBILITY
aspartic proteases	low	flexible
carbonic anhydrases	high	rigid
serine proteases	high	rigid
kinases	high	flexible

Table 7.19: Properties of the four queries.

case, however, certain requirements must be fulfilled and more sophisticated approaches are needed. In particular the used measure should be very flexible, assigning sufficiently high scores or low distances also in the case of a deformation. Especially a low populated enzyme class poses a high challenge for a measure. If the query cavity belongs to such a class the measure has to ensure that the few remaining enzymes of the same class obtain the highest score or the lowest distance among the 186,507 binding sites in the database.

7.4.2 Results

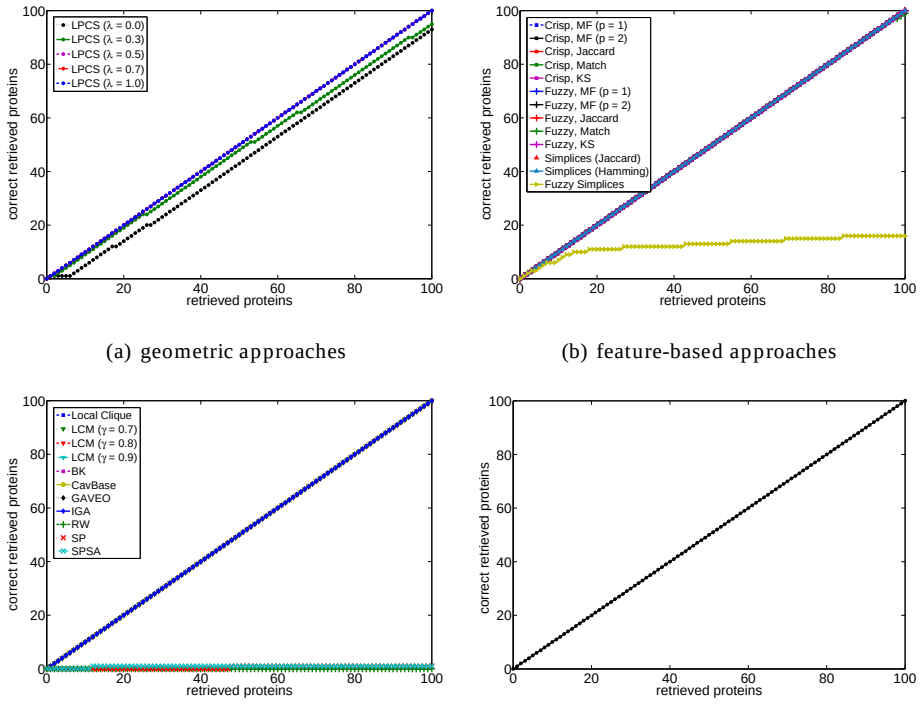
The results on the retrieval experiment will be evaluated in a very intuitive graphical form. There exist different well-known techniques for the evaluation of retrievals, as receiver operating characteristic (ROC) curves, mean average precision or normalized discounted cumulative gain (Manning et al., 2008). However, these state-of-the art approaches are not fully appropriate in the case considered here. In the case of an ROC curve a problem occurs, e.g., if one tries to calculate the *false-positive* (fp) rate: The subset of the database CavBase used in these experiments contains 186,507 structures, many of which are not (fully) EC annotated, so that the calculation of the fp-rate becomes very difficult. A better approach for this specific problem is to consider the retrieved protein structures only. Therefore another evaluation is employed in this thesis which is based on a monotonically increasing $\{0,\ldots,r\} \longrightarrow \{0,\ldots,r\}$ mapping in the case of r retrieved objects. Beginning in $(0,0)$, this function is recursively defined for $i > 0$ as

$$f(i) = \begin{cases} f(i-1)+1 & \text{if the } i\text{-th retrieved object is a correct hit} \\ f(i-1) & \text{otherwise} \end{cases} .$$

Obviously, the described function starts at the origin of the coordinate system and is plotting i against the relevant structures in the top-i. Thus, the identity indicates a perfect ranking, whereas the other extreme, a curve on the x-axis, indicates that all retrieved object are incorrect. In the following, all methods will be evaluated in the retrieval setting described above, where the number r of retrieved structures is set to 100.

Carbonic Anhydrases

As first query the carbonic anhydrase `2eu2` is considered. As shown in Figure 7.26, in each class of representations, there are several measures which perform optimal according to the EC nomenclature. The geometric approaches perform the better, the closer λ is set to 1.0, a setting realizing strict equivalence. The binding site `2eu2.2` is quite small and consists of 59 pseudocenters. A measure which is based on inclusion will search in the database for subsets of pseudocenters which are similar to that small binding site, a procedure which obviously can cause a number of false hits. Feature-based approaches perform surprisingly well. Except the measure based on fuzzy simplices, all remaining



(a) geometric approaches

(b) feature-based approaches

(c) graph-based approaches

(d) sequence-based approaches

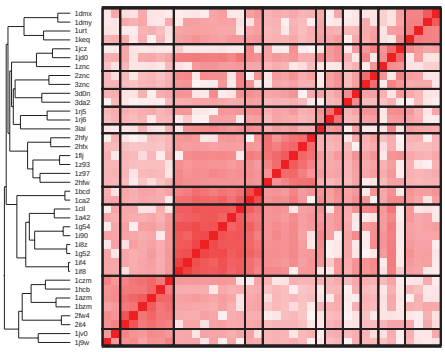
Figure 7.26: Results of the different types of measures on the query `2eu2.1`.

measures of that class perform optimal. The same holds for graph-based approaches, where only measures derived from the R-convolution framework fail completely. As already seen in the classification experiment, the R-convolution framework is not appropriate to be used on larger graphs, e.g. graphs representing protein binding sites. This framework is clearly more suited for comparing small graphs, as graphs representing small molecules or compounds. Sequence-based approaches perform also well on this query and lead – at least according to the evaluation used here – to an optimal result.

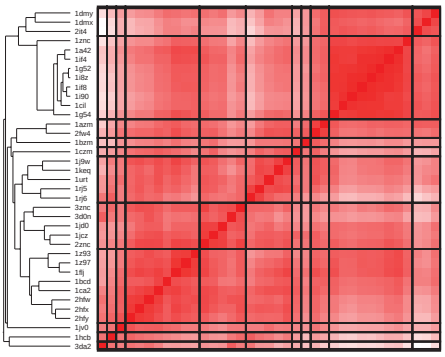
Due to the aforementioned problems caused by the validation criterion based on the EC class only, another experiment is conducted on carbonic anhydrases to demonstrate that structural differences can indeed be recognized. This experiment is taken from (Kuhn et al., 2006), where a set of 37 different carbonic anhydrases was considered. These anhydrases can be divided into different groups, namely CA-I (8 cavities), CA-IIa (8 cavities), CA-IIb (2 cavities), CA-III(6 cavities), CA-IV (3 cavities), CA-V (4 cavities), CA-IX (1 cavity), CA-XII (2 cavities), CA-XIII (2 cavities) and CA-XIV (2 cavities). Under spe-

cial focus are the protein binding sites in 1bcd and 1ca2, as well as 1znc, 2znc and 3znc. The former two cavities belong to CA-II, however, in contrast to all remaining cavities in these examples they differ in the conformation of His64 and exhibit the so-called *out conformation* (Kuhn et al., 2006). For the latter three cavities a general similarity is known, however, they share only a small sequence identity of 56%. Moreover, 2znc and 3znc are fromthe mouse, whereas 1znc orginates from humans and is significantly smaller than their murine counterparts.

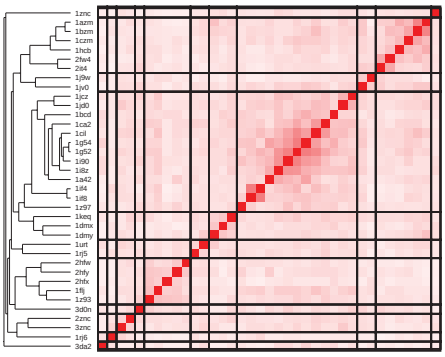
For each representation a measure is taken, namely LPCS ($\lambda = 1.0$) as representative for geometric approaches, the crisp histogram approach in combination with the match distance as representative for feature-based methods, finally the maximum common subgraph calculated with the Bron-Kerbosch algorithm as representative for graph-based methods. Using these three methods, similarity matrices are calculated and passed to the CLUTO clustering toolbox (Karypis, 2006). For clustering the scluto method suitable to be applied on relational data is used with the setting -clmethod=rb -fulltree (repeated bisections and complete hierarchical tree), where the number of clusters is set to 11. The thus calculated clusters are depicted in Figure 7.27, a clustering based on CavBase scores can be taken from (Kuhn et al., 2006), another clustering based on sequence alignment from (Fober et al., 2011). Regarding the cavities 1znc, 2znc and 3znc, only the geometric and graph-based approach are able to group human and murine CA-IVs into individual clusters. The graph-based approach moreover ensures that the human CA-IV is assigned to its own cluster. In the case of a similarity matrix calculated with the feature-based approach, CLUTO cannot form such clusters anymore. However, murine CA-IV cavities are still grouped together, though not in their own separate but instead in a larger cluster with other carbonic anhydrases from different groups. CA-II cavities in out conformation are grouped in an individual cluster only if the geometric approach is used as similarity measure. Using the feature- or graph-based similarity measure, CA-II *out* and *in* conformations cannot be separated anymore and end-up in the same cluster. However, by considering the dendrogram one can recognize that cavities in same conformation are indeed arranged closely. Thus, all three types of representation are able to recognize structural differences and led to a greater or lesser extent to meaningful clusters.



(a) geometric approach



(b) feature-based approach

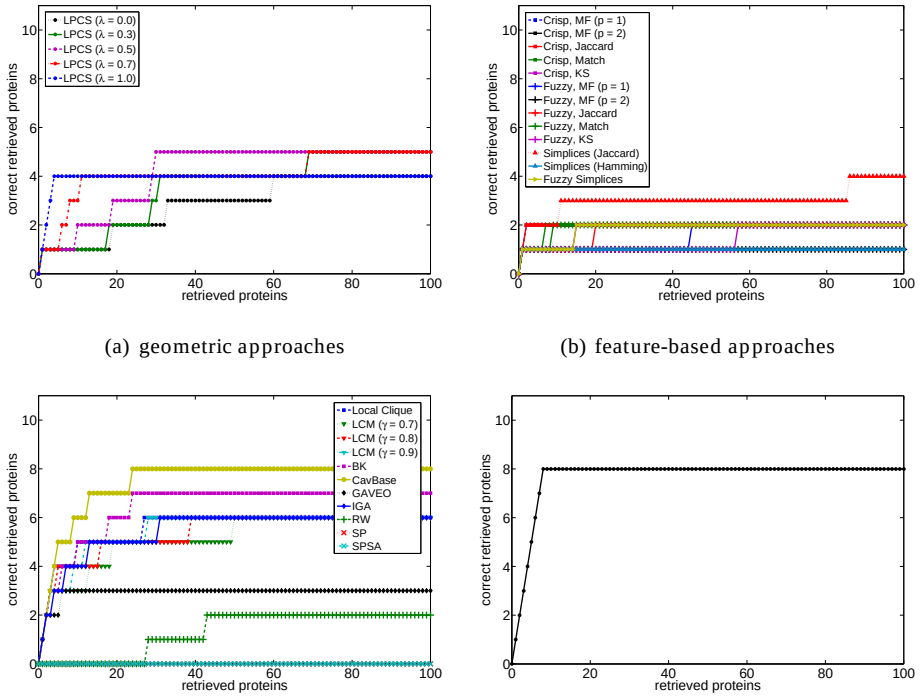


(c) graph-based approach

Figure 7.27: Carbonic anhydrases which were clustered with the CLUTO clustering toolbox; the number of clusters was set by an expert to 11.

Aspartic Protease

In the next experiment the query `1eag.1`, a pocket from the aspartic protease SAP, is considered. Even though there are 11 entries in the PDB which share the same EC number with `1eag`, the CavBase release used in this thesis contains only 8 structures of that EC class. The correctly retrieved structures are visualized in Figure 7.28, where obviously graph-based approaches lead to very good results. The internal CavBase measure is able to retrieve all 8 relevant structures on early ranks, the measure based on the maximum common subgraph misses one structure but can retrieve the remaining 7 cavities reliably. Measures based on the R-convolution framework again turn out to be inappropriate; both SP-kernel variants do not retrieve any correct structure in the top-100. Even though the RW-kernel is able to retrieve two correct structures, they appear only on position 28 and 43. GAVEO which is based on the con-



(a) geometric approaches

(b) feature-based approaches

(c) graph-based approaches

(d) sequence-based approach

Figure 7.28: Results of the different types of measures on the query `1eag.1`.

cept of a graph edit distance performs also poorly. The main issue here are the scoring parameters which must be set optimally. However, a search for optimal parameters becomes infeasible for such a large study. The equivalent based on a greedy heuristic (IGA) exploits the maximum common subgraph and is hence not as sensitive to scoring parameters as GAVEO. Therefore, IGA indeed produces better retrievals. Feature-based methods fail on this query, too. The main problem here is that this class of measure maps protein binding sites of different size onto vectors of fixed length. Hence a certain loss of information appears, in particular it can happen that dissimilar binding sites are mapped onto similar vectors. Hence dissimilar binding sites can become assigned a low distance even tough they are not similar to the query. On the other hand, it cannot be ruled out that the remaining 7 aspartic proteases are mapped onto vectors that differ from the vector of the query due to noise and structural flexibility. Altogether this can lead to a ranking in which irrelevant structures are ranked prior to the relevant ones. Geometric approaches perform good, but not optimal since only 4 relevant structures could be retrieved

reliably. Using a relaxed measure, i.e. a measure not focusing on equivalence

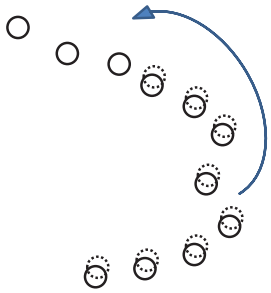
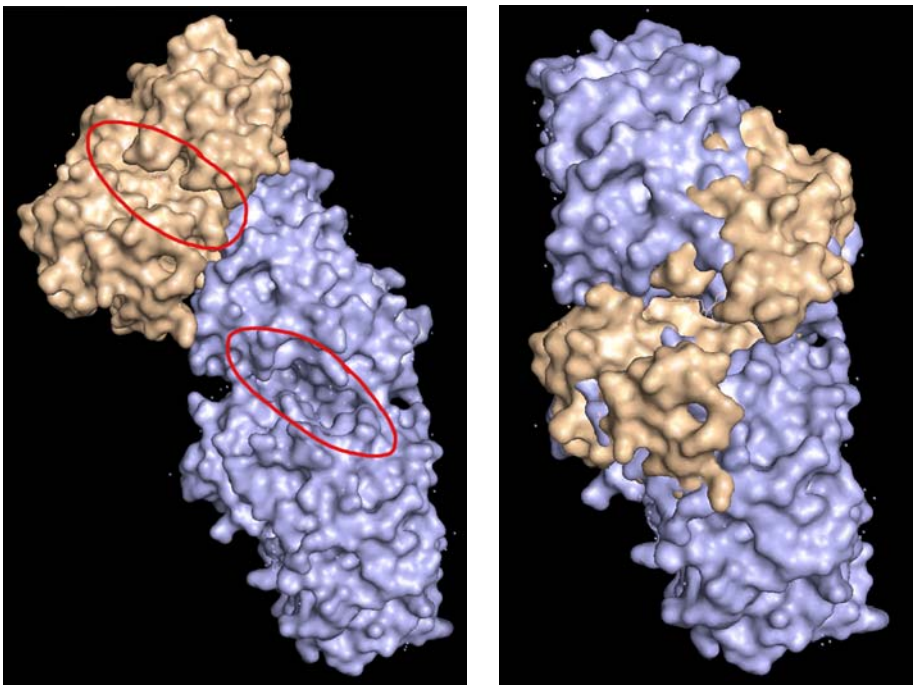


Figure 7.29: Behavior of the labeled point cloud superposition measure on two binding sites (dashed or solid balls, respectively) that differ strongly in the number of pseudocenters. The additional pseudocenters act as a kind attractor and skew the superposition.

but instead on a trade-off between equivalence and inclusion, even 5 relevant structures can be retrieved. A possible reason for the failure of geometric approaches is the inflexible and rigid model. In the case of aspartic proteases the relevant active sites differ strongly in the number of pseudocenters which vary from 88 to 146. Using a relaxation of equivalence can resolve this problem to some degree since partial matches become sufficient to reach the maximum score. However, the additional pseudocenters contained in the larger binding site cause another negative effect: They attract the smaller binding size as illustrated in Figure 7.29 and skew the superposition. This effect is moreover visualized in Figures 7.30 and 7.31, where two binding sites of similar size and two binding sites of significantly different size are superimposed. In the former case a very good superposition is obtained, whereas in the latter case the binding sites are orientated in completely different directions. The sequence-based Smith-Waterman algorithm results with respect to this query in the best proposal. However, one should notice that proteins with a high sequence identity usually share also structural similarity. Since function of a protein is strongly related to its structure, and because evaluation focuses on function, such an optimal result is even expected.

In the last experiment, the calculation of a retrieval is repeated with the geometric approach, however, instead of using `1eag.1` as query, the aspartic protease `1j71.2` is taken, which exhibits the smallest active site among these proteases. Moreover, the geometric measure is parameterized by $\lambda = 0.0$, hence inclusion is considered instead of strict equivalence. Using this setting, the rel-



(a) rotation of 1eag; active sites of 1eag and 2qzx are emphasized by a red circle

(b) rotation and translation of 1eag

Figure 7.30: Superposition of the proteins 1eag and 2qzx. The number of pseudocenters in the corresponding binding sites is more or less equal.

evant structures are placed on rank 1 (1j71), 2 (3fv3), 3 (1zap), 4 (1eag), 5 (2h6s), 10 (2qzw), 82 (2qzx) and 27,879 (2h6t). Hence, the retrieval could be improved by taking the smallest cavity as the query indicating that the above mentioned problem indeed influences the result of the labeled point cloud approach. In practice the latter experimental setting is regularly applied even in an extended fashion: At the Cambridge Crystallographic Data Centre, e.g., researches consider each query very carefully, use their expert knowledge to select the most relevant pseudocenters from the catalytic center of a protein and use only these small set of pseudocenters as query.

Serine Protease

Using the serine protease 2oq5.1 as query leads to the retrieval depicted in Figure 7.32, where once again the sequence-based measure performs optimal due to the aforementioned reasons. Many of the geometric and graph-based approaches lead to optimal results. However, geometric approaches perform

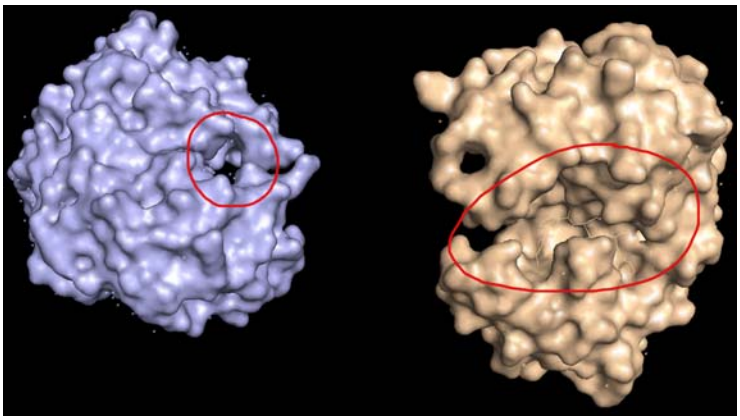
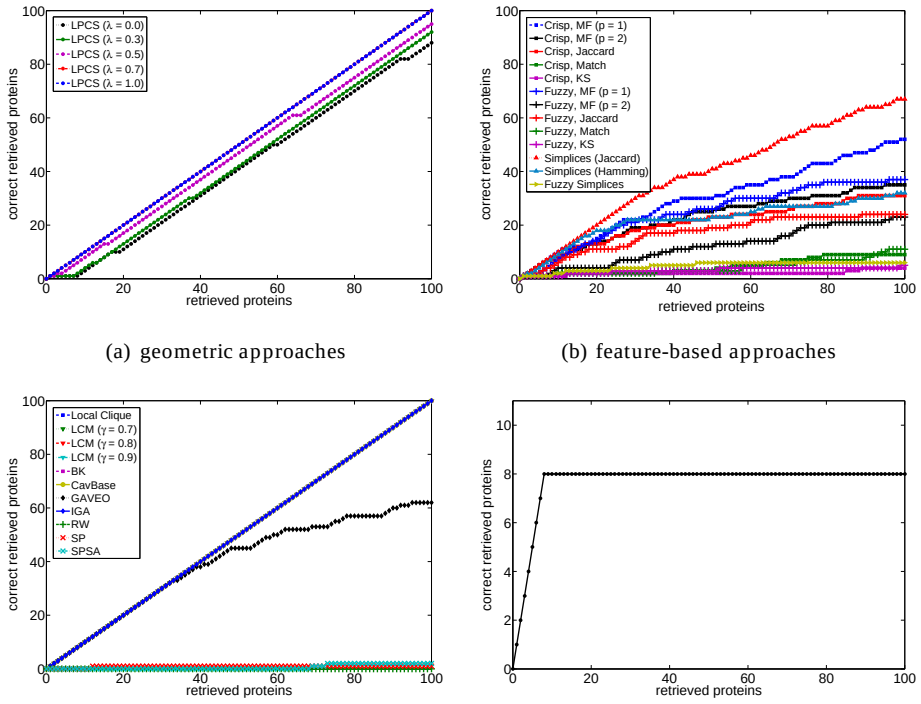


Figure 7.31: Rotation which superimposes the proteins `1eag` and `1j71` optimally according to the LPCS measure. Binding sites of both proteins differ strongly in the number of pseudocenters.

increasingly worse the closer they are to the concept of inclusion. Since serine proteases are known to exhibit rigid binding sites, a rigid measure based on the concept of equivalence can handle these cavities very well. A relaxation based on inclusion will only increase the false-positive rate. Within the class of graph-based measures again graph kernels fail completely. The GAVEO method is also inferior to the remaining measures from this class due to the scoring parameters which influence this measure enormously and which are very hard to adjust in an appropriate manner. Some realizations of the feature-based measure perform as in the case of carbonic anhydrases well, though not optimal. Especially simplices are able to retrieve only relevant structures in the top-30. Obviously this type of feature can capture constellations which are obviously typical for this class of enzymes. 2-dimensional histograms seem not to have the ability to capture such information. As already discussed, these class of measure suffers from the fact that dissimilar cavities can be mapped onto similar vectors, an effect which moreover explains the suboptimal results.

Kinase

Finally kinases are considered, in detail the query pocket from the kinase `3hec.3` for which the retrievals are plotted in Figure 7.33. On this query all three types of measure perform suboptimal: The best geometric instance retrieves 40 relevant structures in the top-100, the best feature-based approach only 20 relevant structures and graph-based approaches at least 60 relevant



(a) geometric approaches

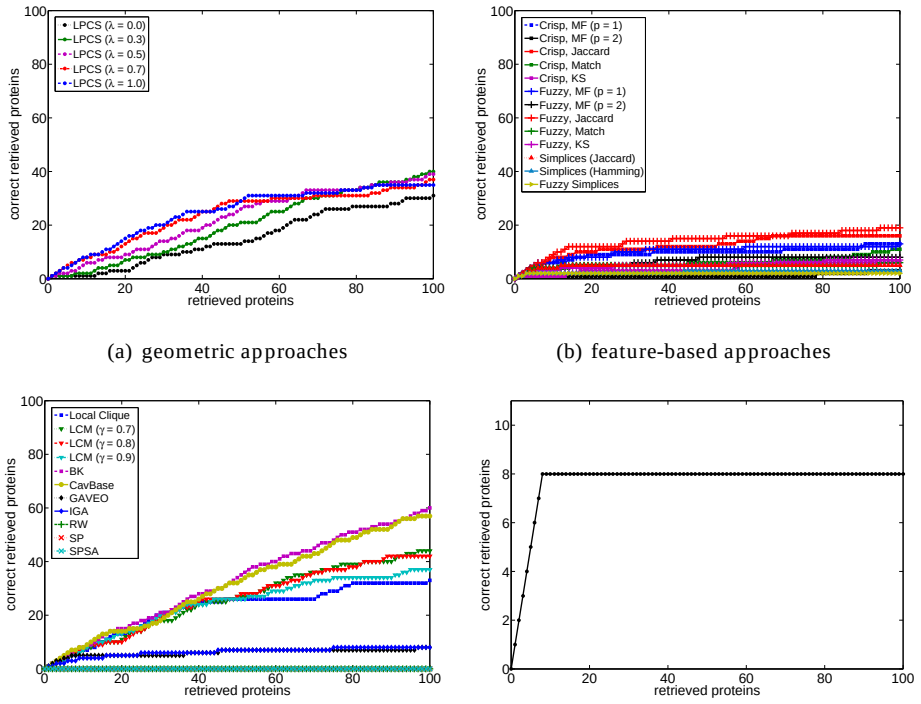
(b) feature-based approaches

(c) graph-based approaches

(d) sequence-based approaches

Figure 7.32: Results of the different types of measures on the query `2oq5.1`.

structures in the top-100. One reason for the bad performance of the measures on that query pocket is its chemical task and recognition properties. Kinases transfer a phosphate group from the nucleoside adenosine triphosphate (ATP), however, ATP as a ligand is recognized by a large variety of very different proteins as substrate or cofactor, among them also other enzymes which do not belong to the EC class of `3hec` but possibly share a considerable similarity in their active sites. Moreover, upon the mechanistic pathway kinases subdivide into *active* and *inactive* configurations which can differ enormously in structure; crystal structures can be determined in the different activation states of the protein, too. Hence, a more careful consideration and visual inspection of the results would be desirable, however, as for the three former queries one should keep in mind that the top-100 lists were generated with 29 different methods. This results in $4 \cdot 29 \cdot 100 = 11,600$ different structures for which it would become necessary to perform a detailed visual inspection. In the context of this Ph.D. thesis which originates from the Mathematics and Computer Science department this task would become pretty complex and costly. Hence, the



(a) geometric approaches

(b) feature-based approaches

(c) graph-based approaches

(d) sequence-based approaches

Figure 7.33: Results of the different types of measures on the query `3hec.3`.

better manageable though not optimal evaluation based on EC numbers was performed. This evaluation again favors the sequence-based approach leading to the corresponding optimal result as depicted in Figure 7.33 (d).

Feature-based Approaches as Filter to Remove Irrelevant Structures

Feature-based approaches perform in general inferior compared to geometric or graph-based approaches. On the other hand, however, they are very efficient and allowing for the precomputation of feature vectors leading afterwards to runtimes significantly below 1 ms needed for a comparison. A way to use them in a more reliable setting is to perform calculations step-wise as illustrated in Figure 7.34, where especially histogram-based approaches are emphasized. Given a query, it is necessary to calculate the (dis)similarities to all other cavities in CavBase. This time-consuming step can be realized by the histogram-based approach which requires approximately 18 minutes on the mean (without using precomputed vectors, cf. Table 7.18). The thus generated distances can be used to discard a subset of protein binding sites of CavBase

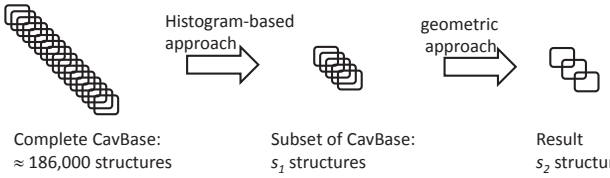
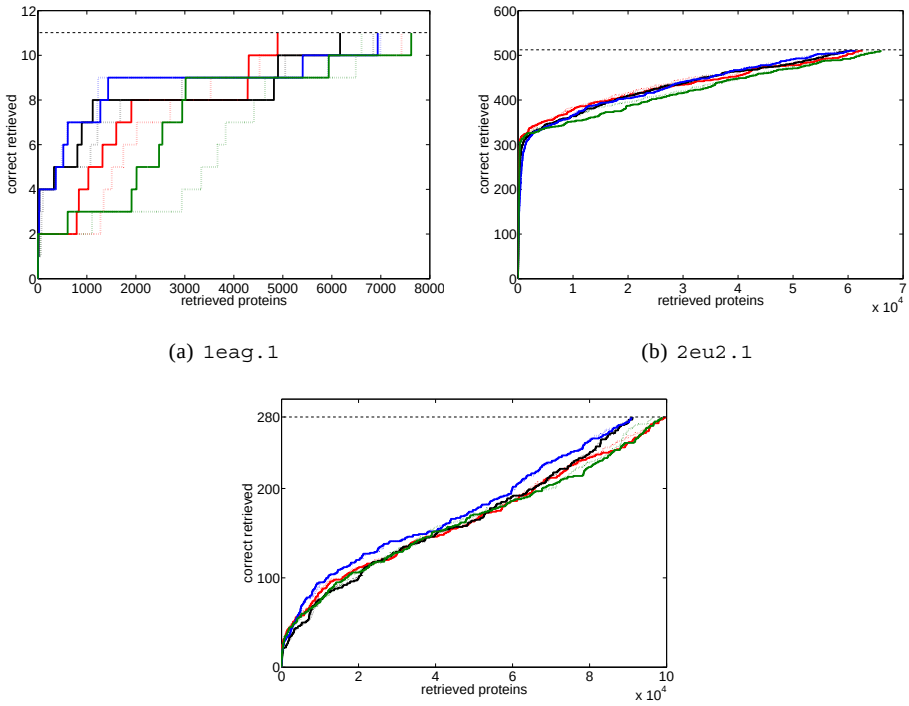


Figure 7.34: Flow used to speed-up the retrieval of similar structures

with an exceptionally high distance eventually leading to a much smaller set of binding sites. On this smaller set, a more precise though computationally more complex approach is applied to obtain another set of (dis)similarities which are finally used to retrieve the r binding sites most similar to the query. The key idea behind this procedure is as follows: Feature-based approaches perform bad because two dissimilar protein binding sites can become assigned a low distance, e.g., if they are mapped onto very similar vectors. Fortunately, the other direction does not hold, i.e., if the distance is high, both proteins must be dissimilar. The proposed approach exploits this property and will discard those binding sites which have exceptionally high distance, thus, leading to a much smaller set which can be afterwards compared by a more exact method.

An important question in this regard is the number of binding sites that can be discarded in the first step. To estimate this number, again the queries `1eag.1`, `2eu2.1` and `3hec.3` are considered. For each query the ground-truth is extracted from the PDB by taking all proteins which belong to the EC class the query exhibits. Afterwards the retrieval is plotted as usual, until all relevant structures are retrieved. The thus generated results are depicted in Figure 7.35 in the form of a distribution of correctly retrieved structures. These distributions indicate that feature-based approaches are indeed appropriate for being used as a filter. If one takes the 20,000 binding sites which have the smallest distance to the query, the resulting set will contain already more than 100 relevant structures. If one is satisfied with 10 relevant structures, already 5,000 binding sites classified by a feature-based approach as most similar to the query would be sufficient. On this strongly reduced sets more exact approaches can be applied to retrieve the final ranking much more efficient. In general it seems that 50% of the binding sites contained in CavBase can be removed by a feature-based approach without loosing any relevant structure.

In terms of runtime the following results are obtained: Using a feature-based approach to reject the aforementioned 50% of binding sits from CavBase



(a) `1eag.1` (b) `2eu2.1`

(c) `3hec.3`

Figure 7.35: Distribution of correctly retrieved structures generated by using histogram-based approaches (distances and properties jointly): blue, black, red and green lines correspond, respectively, to the Minkowski Form distance with parameter $p = 1$ or $p = 2$, the the Jaccard measure and the match distance; solid lines indicate crisp and dashed lines fuzzy approaches.

and by applying on the reduced set afterwards LPCS to get the final ranking will decrease the overall runtime from 61.2 hours to 30.9 hours if one core of an Intel® Core™ i7 860 CPU (2.80 GHz) machine is used; if the CavBase measure is applied instead of LPCS even a reduction from 680.25 to 340.43 hours is obtaind. The final results, however, remain the same. By rejecting more structures in the first step, the runtime can be even reduced further, albeit by increasing the probability of ending-up with a top-*k* list which differes from that list one would obtain without filtering irrelevant structures.

7.4.3 Summary

For the retrieval of structures from a very large database only geometric or graph-based approaches should be applied. Feature-based approaches can

lead to a high loss of information and therefore cannot solve this kind of problem reliable. Geometric approaches perform very well on rigid and highly populated enzyme classes. Here the underlaying rigid model does not pose a serious problem, moreover the local optima do not pose a serious problem. Even in the case that the search is not successful on a relevant structure, there remain enough relevant ones for a good retrieval. In the case of a graph-based representation, especially the approaches based on common subgraphs perform well. This concept does not require the specification of a large set of parameters (influencing the scores), moreover, the methods calculating the common subgraphs are deterministic. Hence, they cannot get stuck in a local optimum.

Feature-based approaches, nevertheless, can be applied in the retrieval setting as well. It was shown that at the end of the ranking these methods produce only irrelevant proteins are placed. Given a query, feature-based approaches can therefore be used to calculate efficiently a ranking of all structures in CavBase. By removing structures with very high distance, the set of remaining structures becomes much smaller. By applying a more reliable geometric or graph-based measure, still good ranking is generated, however, the calculation becomes much more efficient.

Finally, in Figure 7.36 a comparison between all 29 measures is given in terms of a heatmap. To generate this figure, the kinase query `3hec.3` was taken and the top-100 structures were retrieved from CavBase. For this purpose each measure was taken, hence 29 top-100 lists were generated which were subsequently compared in a pairwise manner by the Kendall distance method proposed by Fagin et al. (2003). This procedure leads to a square matrix of size 29 indicating the relationships between measures. Obviously graph-based and geometric measures lead to similar top-100 lists. Moreover it turns out that the top-100 lists generated by the CavBase measure and the geometric measure which focuses on inclusion ($\lambda = 0.0$) are quite similar. Keeping in mind that CavBase is internally also looking for inclusion, this result is reasonable. Even though the top-100 lists generated by feature-based methods share a certain similarity among one another, they are not very similar to the top-100 lists generated with geometric or graph-based approaches. However, especially the histogram approaches which employ bin-by-bin measures share at least a certain similarity with geometric and graph-based methods. This is in accordance to the classification and retrieval results, where these feature-based instances also performed best. Measures which are based on the R-convolution

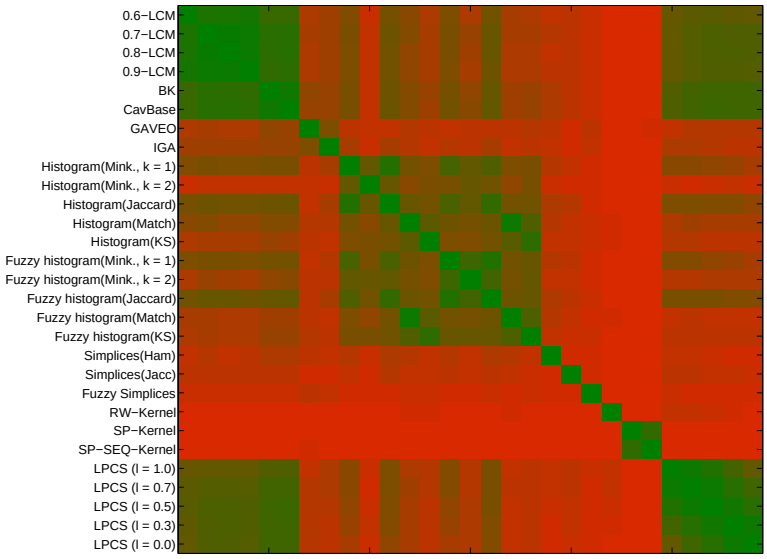


Figure 7.36: Comparison between different measures (cf. Table 7.1) on the query `3HEC.3`; all measures were used to retrieve 100 structures most similar to `3HEC.3`. The thus generated rankings were compared pairwise with the Kendall distance leading to a (29×29) matrix. This matrix is depicted as heatmap which visualizes the distance between different measures from green (= low) to red (= high).

framework share no similarity to the other measures. This result is also supported by the former experiments.

7.5 Multiple Alignments

The last experiment focuses on the construction of multiple alignments. For the calculation of multiple alignments only geometric or graph-based approaches can be used; feature-based methods do not consider single residues, hence, are inappropriate for this purpose. If structures are considered on the geometric level, multiple point cloud alignment (MPCA) can be calculated by means of the geometric alignment algorithm. Even though this approach calculates only pairwise alignments, they nevertheless can be extended to multiple ones by applying merging techniques as tree- or star-alignment. By solving the *m*-partite pivot graph matching problem an MPCA can be calculated even in one step. If a graph-based representation is used, multiple graph alignment (MGA) can be constructed either by using an evolutionary algorithm (GAVEO) or an greedy heuristic (IGA) as proposed by Weskamp et al. (2007).

7.5.1 Dataset

For the calculation of multiple alignments new datasets are required which are known to contain a conserved pattern. Unfortunately, the problem of aligning more than two structures (representing protein binding sites) is not often considered in the literature, hence, only few benchmark sets are available. In the experiment conducted here two of them will be used. The first dataset *benzamidine* does not consist of protein binding sites but contains instead benzamidine derivates. All structures in this dataset contain a clear pattern, hence the hypothetical ground truth is known for this kind of data which allows on the other hand to assess if a method was able to align multiple structures correctly. The second dataset used in this section is called *ATP/ANP* and was proposed by Shatsky (2006). The main advantage of this dataset is that it consists of protein binding sites which were taken directly from CavBase. Moreover it is known that these binding sites contain common patterns. However, one should notice that in contrast to the benzamidine core fragment which should be regarded as the ground truth, the pattern reported by Shatsky (2006) is only the result of his *MultiBind* approach.

Benzamidine

For a first proof-of-concept of the approaches calculating multiple structural alignments, a dataset consisting of 87 compounds is analyzed that belong to a series of selective thrombin inhibitors which were taken from a 3D-QSAR study (Böhm et al., 1999). This dataset is suitable for conducting experiments in a systematic way, as it is quite homogeneous and relatively small, i.e. the descriptors contain 47 – 100 points, where each point corresponds to an atom. Moreover, as the 87 compounds all share a common core fragment which is an amide derivative of benzene consisting of 25 atoms (11 hydrogens), the dataset contains a clear and unambiguous target pattern. Hence, the ground truth is known and can be used to decide if an algorithm returns the correct result. Since the target patterns are distributed over two different regions with a variety of substituents this dataset poses in addition a certain challenge for the algorithms. However, for performing experiments the complete dataset is not used since the multiple alignment problem would become very hard; instead multiple alignments of size 2 – 16 will be constructed, where the structures to align will be selected randomly from the 87 compounds.

ATP/ANP Binding Sites

The ATP/ANP dataset consists of five protein binding sites which were extracted from five different protein kinases, namely cAMP-dependent protein kinase (`1cdk.5`), cyclin-dependent protein kinase (`1hck.3`), glycogen phosphorylase kinase (`1phk.1`), c-Src tyrosine kinase (`2src.3`) and casein kinase-1 (`1csn.1`). Since the number of structures is rather small, it makes no sense to split the dataset further into random subsets. Instead, the whole dataset is processed in one step leading to the problem of aligning 5 protein binding sites simultaneously which exhibit on average 69 pseudocenters. By applying the *MultiBind* algorithm, Shatsky (2006) detected a common pattern of size 13 pseudocenters in this dataset. Even though this pattern should not be regarded as ground truth, the methods proposed in this thesis should find as far as possible a pattern that has at least the same size and a small rmsd value.

7.5.2 Results

By applying the aforementioned algorithms on a certain dataset, an MGA or MPCA is produced which can be used to derive a conserved pattern as follows: An alignment $\mathcal{A}$ consists basically of assignments $a \in \mathcal{A}$ which can be mapped by using the functions (2.5) and (2.6) onto the unit interval $[0,1]$. An assignment is called conserved if the two values are above two thresholds ω and ξ. The union of all conserved assignments is finally called conserved pattern. To decide if an algorithm calculating multiple alignments performs well, the conserved pattern can be compared to the ground truth. A simple 0/1 measure which will be used in this section returns 1 if the ground truth is contained in the conserved pattern and 0 otherwise. If the ground truth is not known, such an approach obviously cannot be used. In this case two quality scores are the size of the detected conserved pattern and its rmsd value.

Benzamidine

From the benzamidine dataset 100 random subsets of c compounds are selected and MPCAs or MGAs are calculated. Since this dataset contains a conserved pattern, the ground truth is known and enables the usage of the 0/1 measure introduced above. Hence, in sum 100 results of type 0/1 are obtained which are summed up and finally divided by 100 leading to the percentage of calculations which were able to align the target pattern correctly. The thus obtained

results are summarized in Table 7.20. Among the graph-based approaches

Table 7.20: Fraction of alignments in which the benzamidine core fragment was fully conserved in alignments of $c = \{2, 4, 8, 16\}$ structures. The parameters ω and ξ were set to 1.0 and 0.9, respectively.

c	2	4	8	16
IGA	0.58	0.38	0.14	0.04
GAVEO	0.97	0.92	0.80	0.76
GAVEO*	0.99	0.96	0.95	0.95
GeoAlign (star)	0.97	0.92	0.80	0.76
GeoAlign (m-partite)	0.93	0.83	0.78	0.67
GeoAlign (tree)	0.97	0.96	0.93	0.90

GAVEO* performs best, IGA clearly worst and GAVEO is in-between. Even though GAVEO is not employing a greedy heuristic the search space on which it is operating is very large and grows super-exponentially with the number of structures one wants to align. Thus it is very likely that GAVEO is not able to finish search in (reasonable) time or that it gets stuck in a local optimum. GAVEO* operates in contrast on a much smaller search space since it is decomposing the multiple alignment problem into several pairwise problems. Hence, the probability for obtaining optimal pairwise alignments, i.e. alignments which correctly map the target patterns, is much higher. Obviously the subsequently applied merging of optimal pairwise alignments to a multiple alignment can hold the aligned target patterns stable. IGA which is using the same approach for decomposing and merging is however using another method to calculate pairwise alignments. This method is based on a greedy heuristic and leads already in the pairwise case to poor results. Merging these suboptimal pairwise alignments to multiple alignments can only worsen the results since already one suboptimal pairwise alignments (of the $c-1$ required pairwise alignments) leads to a suboptimal multiple alignment. The geometric approach is fortunately also able to retrieve the target pattern reliably. Two variants of this approach are using a decomposition technique, too. However, since the constructed pairwise geometric alignments reliably map the target patterns, these patterns are also mapped correctly by the merging techniques. Between these merging techniques some slight differences are recognizable.

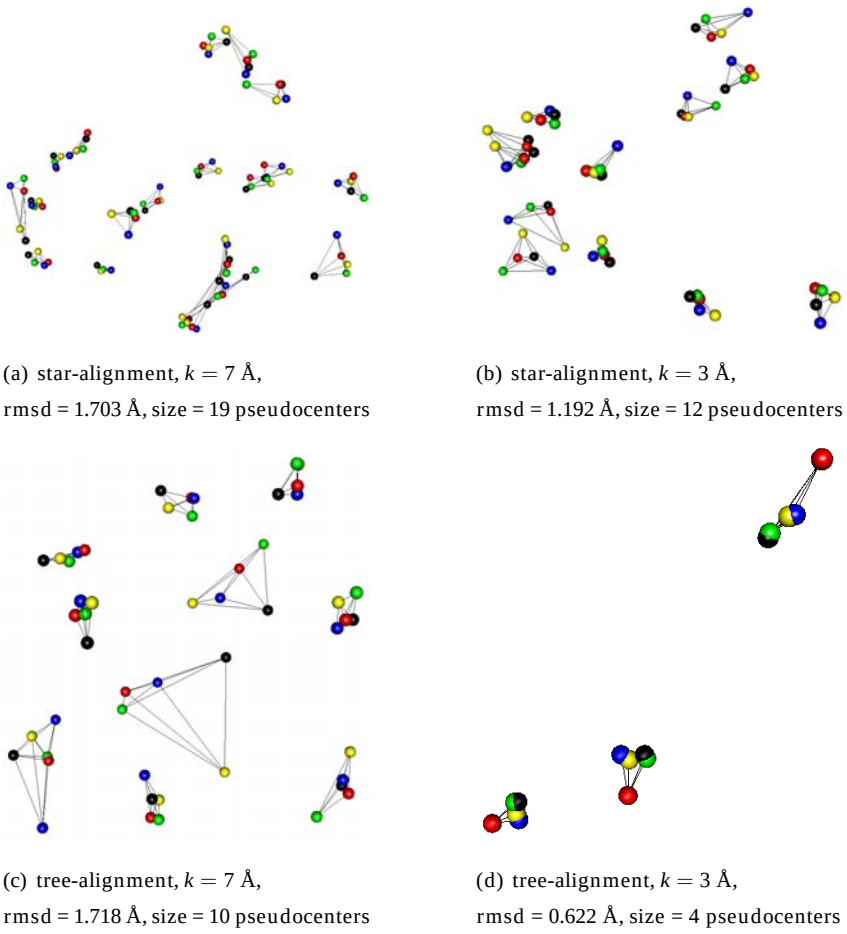
Obviously the tree-alignment approach which exploits information about similarity between the structures leads to the best results. Here more similar structures are merged first, followed by less similar structures. This seems to have benefits compared to a merging based on a pivot-structure which is selected in an greedy way according to a certain scoring system and which exploits no additional information. Solving the MPCA problem in one step by applying m-partite pivot graph matching seems not to be a good solution since the underlying heuristic solver declines even the quality of pairwise alignments.

ATP/ANP Binding Sites

The ATP/ANP dataset contains only 5 protein binding sites, hence, only one multiple alignment of all structures is calculated. As already mentioned, the structures in this dataset share a common pattern which was detected by the *MultiProt* algorithm and exhibits a size of 13 pseudocenters. However, this pattern should not be considered as ground truth, hence, the 0/1 measure cannot be applied. Instead, performance is measured in terms of the size and rmsd value of the found conserved patterns. Since only one run of the methods is required and because some of them employ a randomized algorithm, the experiment is performed as follows: In the case of a deterministic approach (IGA), the algorithm is run exactly once; otherwise 5 runs are performed and the best result in terms of size of the conserved pattern is taken. This pattern is extracted from the multiple alignments by applying the functions (2.5), (2.6) and twice the threshold 1.0. Finally, the conserved pattern is superimposed by means of the Kabsch algorithm (Kabsch, 1976) and visualized by using the `R` RGL package.

Geometric approaches

Multiple geometric alignments can be constructed successfully on the ATP/-ANP dataset by using a 2-stage approach where first pairwise alignments are calculated which are merged subsequently to the multiple alignment. From the thus constructed alignments conserved patterns are extracted which have a size up to 20 pseudocenters and a small rmsd value of 1.703 Å if the star-alignment approach is used for merging and if the parameter k is set to 7 Å. As can be recognized in Figure 7.37 the conserved pattern has in this case indeed a high quality since aligned points are located spatially close. The quality



(a) star-alignment, $k = 7$ Å,
rmsd = 1.703 Å, size = 19 pseudocenters

(b) star-alignment, $k = 3$ Å,
rmsd = 1.192 Å, size = 12 pseudocenters

(c) tree-alignment, $k = 7$ Å,
rmsd = 1.718 Å, size = 10 pseudocenters

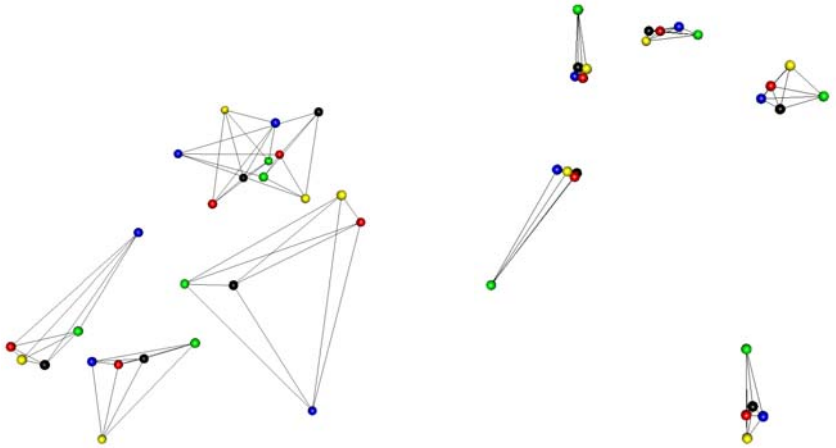
(d) tree-alignment, $k = 3$ Å,
rmsd = 0.622 Å, size = 4 pseudocenters

Figure 7.37: Conserved pattern ($\gamma = 1$, $\xi = 1$) contained in multiple geometric alignment of the structures `1cdk.5` (black), `1hck.3` (red), `1phk.1` (blue), `2src.3` (green) and `1csn.1` (yellow). Merging of pairwise alignments performed respectively with star- or tree-alignment. Straight lines indicate one-to-one correspondences.

in terms of the rmsd value can be increased further by setting k to a smaller value as 3 Å. The parameter k serves as a trade-off between size of the conserved pattern and rmsd value, hence by using $k = 3$ Å a smaller pattern is obtained, however, which matches much better indicated by the low rmsd value of 1.192 Å. Compared to the pattern detected by Shatsky (2006) a clear improvement could be obtained. In the case of tree-alignment which serves as an alternative merging procedure, the results are worse. This is in conflict with the previous experiment where benzamidine derivates were considered and where tree-alignment led to the best results. One should however keep in mind, that benzamidine derivates are quite small and not comparable with the protein

binding sites used in this experiment. Hence, the problem of local optima becomes more serious. Tree-alignment constructs the multiple alignment based on an UPGMA-tree which leads to the calculation of only one alignment. Star-alignment instead is constructing here 5 multiple alignments, calculates for this purpose 20 pairwise alignments and choses finally the best multiple alignment. Therefore the problem of local optima is alleviated which results in improved conserved patterns compared to those constructed by tree-alignment.

In the next experiment still geometric alignments are considered, however, instead of using the 2-stage approach the multiple alignment is calculated in one step by solving the *m*-partite pivot graph matching problem. In the previous experiment in which benzamidine derivates were considered this ap-



(a) *m*-partite, $k = 7$ Å, rmsd = 5.434 Å, size = 5

(b) *m*-partite, $k = 3$ Å, rmsd = 2.037 Å, size = 5

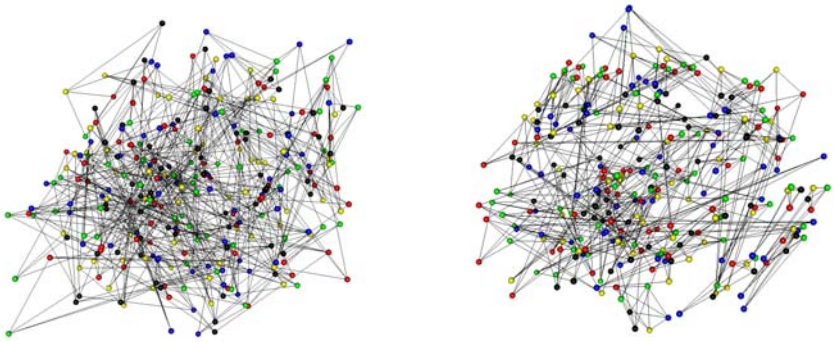
Figure 7.38: Conserved pattern ($\gamma = 1$, $\xi = 1$) contained in multiple geometric alignment of the structures `1cdk.5` (black), `1hck.3` (red), `1phk.1` (blue), `2src.3` (green) and `1csn.1` (yellow). The multiple alignment is calculated at once by solving the *m*-partite pivot graph matching problem. Straight lines indicate one-to-one correspondences.

proach led to the worst result among geometric approaches, a result which is confirmed here. Even though for $k = 3$ Å the conserved pattern contained in the alignment constructed by solving the *m*-partite pivot graph matching exhibits one pseudocenter more compared to the alignment calculated with the tree-alignment approach, its rmsd is nevertheless much higher indicating bad correspondences. Using the setting $k = 7$ Å the conserved pattern is even

much smaller and the rmsd value higher compared to the other geometric approaches.

Graph-based Approaches

In the next experiment graph-based methods are considered which lead without doubts to the largest conserved pattern as can be recognized in Figure 7.39. However, the rmsd value is extremely high, moreover, the long lines used to visualize one-to-one correspondences indicate that only a small amount of spatial information was taken into account during calculation of the multiple alignment and during the extraction of the conserved pattern. A possible reason for this is the used fitness function in combination with its standard parameterization which returns the better scores the more node matches are contained in an alignment. Hence, the optimizers IGA and GAVEO favor an assignment of equally labeled nodes even if such an assignment causes (indirectly) assignments of mismatching edges. The subsequently applied procedure to extract the conserved pattern accesses only the assigned nodes and retrieves such assignments which contain equally labeled nodes of which many exist. Thus the conserved pattern is quite large and consists of 62 pseudocenters, however, the rmsd value is 9.287 Å or 7.490 Å in the case of IGA or GAVEO, respectively. However, the result does not mean that common patterns contained in the 5 binding sites are mapped incorrectly. In fact it is possi-

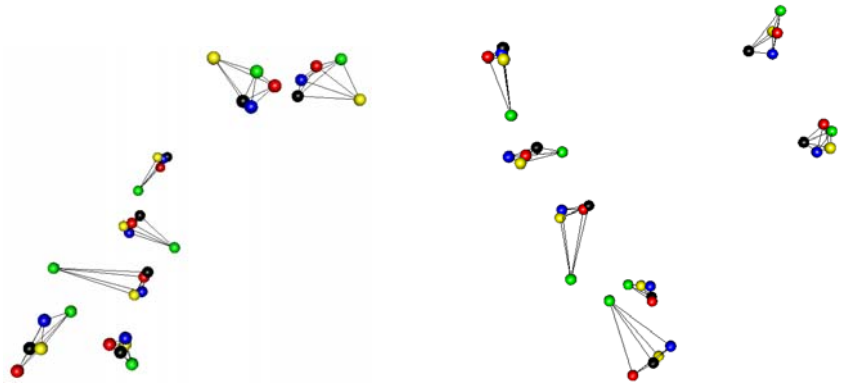


(a) IGA, rmsd = 9.287 Å, size = 62 pseudocenters

(b) GAVEO, rmsd = 7.490 Å, size = 62 pseudocenters

Figure 7.39: Conserved pattern ($\gamma = 1$, $\xi = 1$) contained in multiple graph alignment of the structures `1cdk.5` (black), `1hck.3` (red), `1phk.1` (blue), `2src.3` (green) and `1csn.1` (yellow). Straight lines indicate one-to-one correspondences.

ble that these patterns are mapped correctly, but disappear in Figure 7.39 due to the additional assignments sharing equal label but different geometries. This is even very likely since GAVEO and IGA were able to detect the benzamidine core fragment in the previous experiment reliable. To test an alternative parameterization which focuses stronger on geometry, the parameters given in Table 7.16 are tried in an additional experiment. The thus obtained results are visualized in Figure 7.40 where indeed much smaller conserved patterns are



(a) IGA, rmsd = 1.46 Å, size = 7 pseudocenters

(b) GAVEO, rmsd = 1.46 Å, size = 7 pseudocenters

Figure 7.40: Conserved pattern ($\gamma = 1$, $\xi = 1$) contained in multiple graph alignment of the structures `1cdk.5` (black), `1hck.3` (red), `1phk.1` (blue), `2src.3` (green) and `1csn.1` (yellow). Straight lines indicate one-to-one correspondences. The alternative parameterization given in Table 7.16 and $\epsilon = 2$ Å was used.

detected, which however match much better resulting in a small rmsd value.

7.5.3 Summary

Common patterns in a set of geometric objects such as protein binding sites or chemical compounds can be detected with geometric and graph-based methods as well. In general geometric approaches seem to have a higher ability to find such patterns since they were able to detect the benzamidine core fragment more reliably and because they found larger patterns in the ATP / ANP dataset which exhibit low rmsd values. A main issue of graph-based approaches is the scoring function and its parameterization. Especially in Figures 7.39 and 7.40 the strong dependence on the scoring system is recognizable. Another issue is the method used for the extraction of the conserved patterns which was taken

unchanged from multiple sequence alignment. Hence, the extraction method does not take any geometric information into account and retrieves such assignments which passes the conditions (2.5) and (2.6) that are based only on the labels of the substituents but not on their geometry. As a result many assignments are retrieved which do not match spatially, hence which skew the superpositions calculated by means of the Kabsch algorithm. Especially in the case of graph-based techniques an improved method for the extraction of common patterns would be desirable which is taking geometric information into account to increase the quality of theses patterns in a post-processing step.

8

Conclusion and Outlook

In this thesis measures for the comparison of protein binding sites as well as methods for the calculation of (multiple) alignments were proposed that will be summarized in this chapter. Of course, this thesis could not cover all issues so that interesting extensions will be discussed in the second section of this chapter.

8.1 Conclusion

Starting point of this thesis was CavBase, a data bank for storing protein binding sites. This databank provides a method for the calculation of similarity between these binding sites, moreover, it contains a greedy heuristic[1] for the calculation of multiple alignments. Calculation of similarity between two binding sites is realized in CavBase as follows: The binding sites are transformed into graphs and the largest 100 common subgraphs are detected, each of which is subsequently used to calculate the optimal superposition of the protein binding sites. For each of the determined superpositions, the surface points are finally considered and superimposed leading to a similarity value. To obtain the final score, the maximum is taken from the set of 100 similarity values. Multiple alignments are also constructed by transforming the binding sites into graphs and by detecting the largest 100 common subgraphs, each of which defines a partial pairwise alignment. These pairwise alignments are extended to complete ones by applying a greedy heuristic. Finally, to obtain the multiple alignment, pairwise alignments are merged by the star-alignment heuristic and the best solution from the set of constructed alignments is returned.

[1] This heuristic is not contained in the latest release of CavBase.

One goal of this thesis was to investigate if representations different from graphs also can be used to process on protein binding sites. On the one hand, geometric approaches were proposed which allow to process CavBase data directly, hence, without causing a loss of information as it is the case when using graphs. On the other hand, if one accepts to cause a higher loss of information, protein binding sites can be mapped onto vectors which can be compared afterwards very efficiently. For all three representations — geometric, feature-based and graph-based — methods were introduced which allow space and time efficient calculation of similarity and alignments.

8.1.1 Geometric Approaches

Geometric approaches are characterized by their space efficiency. For the calculation of similarity the space complexity grows linearly with the number of pseudocenters contained in the binding sites. Thus, space does not pose a bottleneck during calculations (as it is the case, e.g., when processing on graphs). Space complexity is increased if alignments have to be constructed but the complexity is still of low polynomial order. The runtime complexity is hard to estimate since an evolutionary algorithm is employed internally. However, the optimization problem to solve is continuous and exhibits in each point the direction of steepest descend, hence, poses no serious problem for optimizers. Having calculated the optimal superpositions, the problem of multiple alignment can be solved in polynomial time.

Regarding the results obtained, the superposition of labeled point clouds led to the highest classification rates and very good retrievals. Obviously, the rigidity of the proposed method did not pose a serious problem, in fact, the afterwards constructed pairwise alignments even indicated that reasonable superpositions and alignments can be calculated even on such a representation. Conserved patterns derived from multiple alignments of randomly chosen subsets of benzamidine derivates and on a further set of binding sites finally indicated that even the construction of multiple alignments works proper.

8.1.2 Feature-based Approaches

Feature-based approaches exhibit a central advantage compared to the other classes. Even though the mapping from protein binding site to such a representation is more complex than, e.g., the mapping onto graphs, the subsequent

comparisons can be performed very efficiently. Moreover, this mapping must be performed only once and can be reused several times. Hence, such approaches are appropriate to be used in very large studies. The price to pay for this efficiency is a high loss of information, since in the general case the mapping back from feature-based representation to binding site is impossible.

The results obtained are conflicting since on the binary classification problem this class of measures performed surprisingly well, whereas on the retrieval setting and the additionally conducted clustering experiment very poor. These differences are caused by the high degree of loss of information. This problem is not very serious if small datasets are considered, as the ATP / NADH dataset which contains 355 binding sites. Here the probability for the mapping of two dissimilar binding sites onto similar feature vectors is very low. When considering 186,507 binding sites this probability increases strongly, hence dissimilar structures mistakenly can be top-ranked. However, the other direction does not hold, i.e., binding sites which are mapped onto different vectors must be dissimilar. This property can be used to reduce the number of structures, hence, to accelerate calculations by first removing a large set of "false" structures and applying a more exact method afterwards. Applied on the 4 queries used in this thesis a significant speed-up could be obtained by using this 2-stage approach. For the calculation of multiple alignments, however, feature-based approaches are obviously inappropriate since this representation does not consider individual residues.

8.1.3 Graph-based Approaches

Protein binding sites represented by graphs become invariant to translation and rotation. This simplifies calculations since the determination of a common coordinate system becomes obsolete. On the other hand, however, the resulting algorithmic problems often become NP-hard. On small inputs such problems can be solved efficiently but become very inefficient on large inputs. Another issue is the space complexity since space required to store graphs grows quadratically with the number of nodes, moreover, often the product graph is employed during calculations which again has a quadratic space complexity. Hence, measures based on a graph representation scale very bad with size of binding sites. Nevertheless, efficient heuristics were proposed in this thesis which seem to be a good alternative to exact though inefficient measures.

Measures which derive similarity from the (approximate) maximum common subgraph perform very well, in the retrieval setting even best. One advantage these measures exhibit is that calculations are deterministic and almost parameterless. Moreover, determining similarity according to the size of the largest common substructure seems also to be reasonable. The CavBase measure is not superior to the novel methods developed here, neither in classification nor in retrieval. This is astonishing, especially because this measure is using much more information. A possible reason might be that another normalization is preformed based on the concept of inclusion. However, this concept works only proper if the pattern one is looking for is selected very carefully. At *The Cambridge Crystallographic Data Centre*, e.g., this pattern is selected by a human who is taking a lot of expert knowledge into account. One type of efficient measures on graphs, so called graph kernels, failed completely on protein binding sites. An obvious drawback of these measures is the all-versus-all comparison of substructures which becomes the more questionable the larger the graphs become. Measures based on the graph edit distance did not perform well, either. The central problem these measures exhibit is their flexibility which makes it very hard to adjust the measure onto a certain task. However, for the calculation of (multiple) alignments the concept of a graph edit distance performed very well since conserved patterns could be mapped correctly independent of the chosen parameterization.

8.1.4 Overall Results

It is very hard to draw a conclusion which of the approaches is best appropriate as measure on protein binding sites. All representations have pros and cons. If one wants to process on large binding sites, graph-based approaches will often fail since the space they require becomes too large. Feature-based and geometric approaches on the other hand are more efficient in this regard and scale much better with the size of the binding sites. However, the former lead to a loss of information whereas the latter suffer from the optimizer internally used which is randomized and does not guarantee the optimal solution. Accordingly, graph-based approaches performed best in the retrieval setting where a very large number of protein binding sites was considered. On smaller datasets, as the ATP / NADH set, feature-based approaches performed astonishingly well, geometric methods even best. Here the probability that

dissimilar binding sites are mapped onto similar feature-vectors is strongly decreased; the problem of calculation of suboptimal scores (e.g., due to local optima) is also not serious, since one suboptimal result can be corrected by another optimal one.

Regarding the calculation of alignments only graph-based and geometric approaches come into consideration, where geometric approaches still scale much better than graph-based methods. Both types on methods were able to map a known conserved pattern correctly, graph-based approaches could solve this task even slightly better. A central problem of graph-based methods is however, that they are very sensitive to the chosen scoring parameters and that they process geometry only indirectly. In this regard, geometric approaches have clearly advantages.

Even though exhibiting some drawbacks (as loss of information), feature-based approaches seem to be a very efficient alternative for the calculation of similarity. Their main drawback is that dissimilar binding sites can be mapped onto similar vectors. However, the other direction does not hold, i.e., dissimilar vectors are derived from dissimilar binding sites. This property can be used in a first step of a study to remove all false-positives in a very efficient way. Afterward more exact methods can be applied which can be based either on a geometric or graph-based representation, dependent on the size of the structures to compare.

However, one should keep in mind that for validation only a small set of experiments was conducted, i.e. one dataset was used for classification, 4 queries for retrieval and 2 datasets for the construction of (multiple) alignments. Hence, it is questionable if one can reliably generalize the results obtained. For that reason it is planed to implement a toolbox containing all algorithms collected in this thesis and to hand over this toolbox to the local department of pharmacy for a long term study.

8.2 Outlook

A large amount of work was spend to improve the similarity retrieval in CavBase in terms of efficiency and quality, and to evaluate if representations different from graphs can be also used to process CavBase data. However, there are still some possibilities for a further improvement which are discussed in this section. This section starts with possible modifications that could lead to

improved similarities, followed by modifications of algorithms that could accelerate calculations. Finally, some applications outside biology are presented for which the methods collected in this thesis could also prove appropriate.

8.2.1 Improvement of Measures

The measure used in CavBase is algorithmically based on subgraph isomorphism. However, it is not purely based on the size of the obtained maximum common subgraph. Instead, a set of common subgraphs is calculated by the Bron-Kerbosch algorithm and further steps are applied to improve the obtained scores (cf. Section 1.2). All methods returning a (partial) alignment can be used directly in this framework, hence, the iterative graph alignment, graph alignment via evolutionary optimization, local clique heuristic and local clique merging as well as the geometric alignment approach. In CavBase the Bron-Kerbosch algorithm simply must be replaced by one of the methods mentioned above, to realize this modification. In the case of labeled point cloud superposition the optimal superposition is immediately available, hence, the inclusion of this approach in CavBase is even more simple since calculation of product graph and application of the Kabsch algorithm become obsolete. However, one should not expect a high improvement since it was shown in this thesis, that the additional steps performed in CavBase lead – if any – only to a slight improvement. Methods calculating raw similarities obviously do not allow for applying these additional steps since they do not return an alignment or a superposition. This concerns all feature-based measures and measures based on the R-convolution framework.

The labeled point cloud superposition approach returns only one optimal superposition, even though there can exist many others exhibiting almost the same score. To enable detection of all appropriate superpositions a search for all local optima can be applied instead of a global optimizer. Here specialized optimization techniques can be applied to detect all local optima, each of which is giving a superposition that can be reused, e.g., in the CavBase framework to superimpose the surface points.

Approaches based on the graph edit distance require specification of a scoring function which should be subject to an investigation since it is questionable if the current realization is optimal. Even though the scoring function offers a high degree of flexibility, their parameterization often poses a challenge. More-

over, the additive combination of a parameterized edge and a parameterized node score has some drawbacks. A possible extension is either to optimize each score separately and to combine both values after appropriate normalization to the overall score, or the multicriterial optimization of both scores. Another idea is to replace the scoring function. Even though needed during calculations, the final distance between two binding sites can be derived from another measure, e.g., the widely used and accepted rmsd value.

As already mentioned, feature-based methods are very efficient, a property which renders them applicable on a more detailed representation, e.g., based on the surface patches of a binding site. Moreover, it would be desirable to have a set of other features which capture global information, as the degree of curvature, the volume and the surface area. A further interesting extension is the incorporation of user knowledge by hand-designed features. A drawback of feature-based methods presented in this thesis is that the set of features is very high. Here feature selection techniques can be applied to enable a more compact and more appropriate vector representation of protein binding sites.

Another way to increase the quality of measures is to improve the pseudocenter representation, hence, the second level of the CavBase loop (cf. Figure 1.2). Related with this idea some work was already done, e.g., in Kuhn et al. (2006) and Krotzky (2011), where pseudocenters were enriched with additional information. On the first level of this loop there are also possibilities for an improvement. In fact, the LigSite algorithm applied on this level comes with some problems, e.g., with the reliable identification of the boundary of a protein binding site. Here novel enhanced techniques could be developed to increase performance even more.

8.2.2 Improvement of Algorithms

Regarding the algorithms there is also potential for an improvement. As already mentioned, the mutation operator of GAVEO is quite simple, however, led so far to best results. However, in case of the evolutionary construction of graph alignments the mutation operator is the best starting point for an improvement. Instead of performing random mutations a local search procedure can be employed. Moreover, instead of allowing any possible transformation one can restrict the search space to such solutions which are observable in the 3-dimensional Euclidean space leading to a strong reduction of the search space.

The bottle-neck of the labeled point cloud superposition method is obviously the nearest neighbor search. This search is realized by a simple approach since the data structures tried so far were not able to increase efficiency due to a relatively small number of points and a high overhead of these data structures. Thus, the break-even-point could not be reached, however, due to the potential a more efficient nearest neighbor search has, improved methods should be developed. A further improvement of the LPCS method would be to replace the fitness function. Currently this function does not allow for the application of an indirect search method, which however could increase efficiency enormously compared to the direct search approach employed in this thesis.

The CavBase method and the measures based on maximum common subgraphs employ internally the Bron-Kerbosch algorithm. This algorithm is designed for finding all cliques in general graphs. The graphs we consider represent, however, geometric structures. Here an interesting question is, if for this type of graph more efficient approaches can be applied. Furthermore, a combination of the graph-based and geometric representation would be desirable which could lead beside the win of information to a win of efficiency.

Some of the novel measures are still too inefficient to perform a large scale study on CavBase data. To accelerate calculations, one can make use of Clusters as done in the experiments performed in this thesis. The problem of using clusters is, however, the limited number of nodes. On general-purpose graphic processor units (GPGPU) the number of cores is not anymore an issue since such units can contain more than 3000 cores. Especially, if simple though many calculations must be performed, as in the case of an evolutionary algorithm where many individuals need to be evaluated, GPGPUs would lead to a dramatical reduction of runtime. In the best case, a GPGPU allows to realize calculation of complexity $\mathcal{O}(n)$ in time $\mathcal{O}(1)$. Meanwhile, there exists a cooperation with the local *Distributed Systems* research group, realizing some algorithms to calculate similarity between protein binding sites on a GPGPU using *OpenCL* (Munshi, 2011).

8.2.3 Applications beside Structural Bioinformatics

The usage of the methods developed in this thesis is not restricted to the comparison of protein binding sites. In fact, all presented algorithms are general-purpose methods, which allow for comparison of arbitrary graphs and point

clouds. As one can imagine, the number of domains in which relational or geometric data is available is very large. Examples are social networks, optical character recognition, fault detection in manufactured parts, image processing and so on.

First experiments were already performed in an image retrieval setting, where two sets of pictures were given leading to a binary classification problem. Each picture was segmented into patches of certain color, where a small discrete set of colors was used. Subsequently, the center of gravity was calculated for each segment and represented by a labeled node. Edges were used to connect nodes which were weighted with the Euclidean distance between the centers of gravity. On the thus created graphs the GAVEO approach was applied leading to a very good classification rate. Moreover, by detecting conserved patterns the identification of common objects in a set of pictures becomes possible. Here also very good results were obtained, hence, even an unsupervised learning task was solved successfully on a set of pictures.

Bibliography

Agrawal, R. and Srikant, R. (1994). Fast Algorithms for Mining Association Rules. In *Conference on Very Large Data Bases*, volume 1215, pages 487–499, Santiago, Chile.

Alexandrov, N. and Fischer, D. (1996). Analysis of Topological and Nontopological Structural Similarities in the PDB: New Examples With Old Structures. *Proteins*, 25(3):354–365.

Alt, H., Behrends, B., and Blömer, J. (1991). Approximate matching of polygonal shapes. In *7th Annual ACM Symposium on Computational Geometry*, pages 186–193, North Conway, USA.

Alt, H. and Guibas, L. J. (1996). Discrete geometric shapes: Matching, interpolation, and approximation: A survey. Technical report, Handbook of Computational Geometry.

Alt, H., Mehlhorn, K., Wagener, H., and Welzl, E. (1988). Congruence, similarity and symmetries of geometric objects. *Discrete Computational Geometry*, 3:237–256.

Altschul, S. F. and Lipman, D. (1989). Trees, stars and multiple biological sequence alignment. *SIAM Journal Applied Mathematics*, 49(1):197–209.

An, J., Totrov, M., and Abagyan, R. (2004). Comprehensive identification of "druggable" protein ligand binding sites. *Genome Informatics*, 15(2):31–41.

Anderson, M. J. and Whitcomb, P. J. (1974). *Design of Experiments*. John Wiley & Sons, Inc., New York, USA.

Arkin, E. M., Kedem, K., Mitchell, J. S. B., Sprinzak, J., and Werman, M. (1992). Matching points into pairwise-disjoint noise regions: Combinatorial bounds and algorithms. *ORSA Journal of Computing*, 4(4):375–386.

Artymiuk, P., Poirrette, A., Grindley, H., Rice, D., and Willett, P. (1994). A Graph-theoretic Approach to the Identification of Three-dimensional Patterns of Amino Acid Side-chains in Protein Structures. *Journal of Molecular Biology*, 243(2):327–344.

Atkinson, M. D. (1987). An optimal algorithm for geometrical congruence. *Journal of Algorithms*, 8:159–172.

Bach, F. R. (2008). Graph kernels between point clouds. In *International Conference on Machine Learning*, pages 25–32, Helsinki, Finland.

Bachar, O., Fischer, D., Nussinov, R., and Wolfson, H. (1993). A computer vision based technique for 3-d sequence-independent structural comparison of proteins. *Protein Engineering*, 6(3):279–287.

Bartz-Beielstein, T., Lasarczyk, C. W. G., and Preuss, M. (2005). Sequential parameter optimization. In *Congress on Evolutionary Computation*, pages 773–780, Edinburgh, Scotland.

Berg, J. and Lässig, M. (2004). Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(41):14689–14694.

Bergamini, P., Cinque, L., Cross, A., Hancock, E., Levialdi, S., and Myers, R. (2000). Efficient Alignment and Correspondence using Edit Distance. In *IAPR International Workshops on Advances in Pattern Recognition*, pages 246–255.

Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York.

Böckenhauer, H.-J. and Bongartz, D. (2007). *Algorithmic Aspects of Bioinformatics*. Springer, Heidelberg, Germany.

Böhm, M., Stürzebecher, J., and Klebe, G. (1999). Three-dimensional quantitative structure-activity relationship analyses using comparative molecular field analysis and comparative molecular similarity indices analysis to elucidate selectivity differences of inhibitors binding to trypsin, thrombin, and factor xa. *Journal of Medicinal Chemistry*, 42(3):458–477.

Borgelt, C. and Berthold, M. (2002). Mining Molecular Fragments: Finding Relevant Substructures of Molecules. In *International Conference on Data Mining*.

Borgelt, C., Meinl, T., and Berthold, M. (2005). MoSS: A Program for Molecular Substructure Mining. In *KDD international workshop on open source data mining: frequent pattern mining implementations*, pages 6–15.

Borgwardt, K., Ong, C., Schonauer, S., Vishwanathan, S., Smola, A., and Kriegel, H. (2005). Protein Function Prediction via Graph Kernels. *Bioinformatics*, 21(1):i47–i56.

Borgwardt, K. M. (2007). *Graph Kernels*. PhD thesis, Ludwig-Maximilians-Universität München, Germany.

Borgwardt, K. M. and Kriegel, H. P. (2005). Shortest-path kernels on graphs. In *International Conference on Data Mining*, pages 74–81, Houston, Texas.

Boukhris, I., Elouedi, Z., Fober, T., Mernberger, M., and Hüllermeier, E. (2009). Similarity analysis of protein binding sites: A generalization of the maximum common subgraph measure based on quasi-clique detection. In *International Conference on Intelligent Systems Design and Applications*, pages 1245–1250, Pisa, Italy.

Bredel, M. and Jacoby, E. (2004). Chemogenomics: an emerging strategy for rapid target and drug discovery. *Nature Reviews Genetics*, 5:262–275.

Brille, P. (2005). A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337:217–239.

Bron, C. and Kerbosch, J. (1973). Finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575 – 577.

Bronstein, I. N., Semendjajew, K. A., Musiol, G., and Muehlig, H. (2008). *Taschenbuch der Mathematik*. Harri Deutsch, Frankfurt am Main, 7 edition.

Bruno, I. J., Cole, J. C., Lommerse, J. P. M., Rowland, R. S., Taylor, R., and Verdonk, M. L. (1997). Isostar: A library of information about nonbonded interactions. *Journal of Computer-Aided Molecular Design*, 11(6):525–537.

Bunke, H. (1999). Error Correcting Graph Matching: On the Influence of the Underlying Cost Function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):917–922.

Bunke, H. and Allermann, G. (1983). Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1:245–253.

Bunke, H. and Bühler, U. (1993). Applications of approxiate string matching to 2D shape recognition. *Pattern Recognition*, 26(12):1797–1812.

Bunke, H. and Jiang, X. (2000). Graph matching and similarity. *Intelligent systems and interfaces*, 15:281 – 304.

Bunke, H., Jiang, X., and Kandel, A. (2000). On the Minimum Common Supergraph of two Graphs. *Computing*, 65(1):13–25.

Bunke, H. and Shearer, K. (1998). A Graph Distance Metric Based on the Maximal Common Subgraph. *Pattern Recognition Letters*, 19(3-4):255–259.

Chalk, A. J., Worth, C. L., Overington, J. P., and Chan, A. W. E. (2004). PDBLIG: Classification of small molecular protein binding in the protein data bank. *Journal of Medical Chemistry*, 47(15):3807–3816.

Chao, K. and Zhang, L. (2009). *Sequence Comparison*. Springer, Heidelberg, Germany.

Christmas, W., Kittler, J., and Petrou, M. (1995). Structural Matching in Computer Vision using Probabilistic Relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):749–764.

Corman, T. H., Stein, C., Leiserson, C. E., and Rivest, R. L. (2001). *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts.

Davidson, E. H., Rast, J. P., Oliveri, P., Ransick, A., Calestani, C., Yuh, C.-H., Minokawa, T., Amore, G., Hinman, V., Arenas-Mena, C., Otim, O., Brown, C. T., Livi, C. B., Lee, P. Y., Revilla, R., Rust, A. G., Pan, Z., Schilstra, M. J., Clarke, P. J. C., Arnone, M. I., Rowen, L., Cameron, R. A., McClay, D. R., Hood, L., and Bolouri, H. (2002). A genomic regulatory network for development. *Science*, 295(5560):1669–1678.

de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2000). *Computational Geometry*. Springer, New York.

Dehaspe, L., Toivonen, H., and King, R. (1998). Finding Frequent Substructures in Chemical Compounds. In *4th International Conference on Knowledge Discovery and Data Mining*, pages 30–36. AAAI Press.

Deza, M. M. and Deza, E. (2009). *Encyclopedia of Distances*. Springer, Heidelberg, Germany.

Dror, O., Benyamini, H., Nussinov, R., and Wolfson, H. (2003). MASS: Multiple Structural Alignment by Secondary Structures. *Bioinformatics*, 19(1):i95–104.

Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81.

Efrat, A. and Itai, A. (1996). Inprovements on bottleneck matching and related problems using geometry. In *12th Annual ACM Symposium on Computational Geometry*, pages 301–310, Philadelphia, Pennsylvania, USA.

Ekins, S. (2004). Predicting undesirable drug interactions with promiscuous proteins in silico. *Drug Discovery Today*, 9(6):276–285.

Fabio, R. (2003). From point cloud to surface: The modeling and vosualization problem. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 24–28, Tarasp-Vulpera, Switzerland.

Fagin, R., Kumar, R., and Sivakumar, D. (2003). Comparing top *k* lists. *SIAM Jounal on Discrete Mathematics*, 17(1):134–160.

Falicov, A. and Cohen, F. (1996). A Surface of Minimum Area Metric for the Structural Comparison of Proteins. *Journal of Molecular Biology*, 258(5):871–892.

Fernández, M. L. and Valiente, G. (2001). A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22(6-7):753 – 758.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345.

Fober, T., Glinca, S., Klebe, G., and Hüllermeier, E. (2011). Superposition and alignment of labeled point clouds. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(6):1653–1666.

Fober, T. and Hüllermeier, E. (2009a). Fuzzy modeling of labeled point cloud superposition for the comparison of protein binding sites. In *IFSA World Congress, EUSFLAT World Conference*, pages 1299–1304, Lisboa, Portugal.

Fober, T. and Hüllermeier, E. (2009b). Multiple geometrical alignments for the analysis of protein active sites. In *19. Workshop Computational Intelligence*, pages 237–255, Dortmund, Germany.

Fober, T. and Hüllermeier, E. (2010). Similarity measures for protein structures based on fuzzy histogram comparison. In *World Congress on Computational Intelligence*, pages 2808–2814, Barcelona, Spain.

Fober, T. and Hüllermeier, E. (2011). Local clique merging: An extension of the maximum common subgraph measure for the classification of graph structures. In *Joint Conference of the German Classification Society and the German Association for Pattern Recognition*, Frankfurt a.M., Germany.

Fober, T., Hüllermeier, E., and Mernberger, M. (2007). Evolutionary construction of multiple graph alignments for the structural analysis of biomolecules. In Mikut, R. and Reischl, M., editors, *Proceedings 17. Workshop Computational Intelligence*, pages 1–14, Witten-Bommerholz, Germany.

Fober, T., Klebe, G., and Hüllermeier, E. (2009a). Efficient construction of multiple geometrical alignments for the comparison of protein binding sites. In *International Conference on Intelligent Systems Design and Applications*, pages 1251–1256, Pisa, Italy.

Fober, T., Mernberger, M., Klebe, G., and Hüllermeier, E. (2009b). Evolutionary construction of multiple graph alignments for the structural analysis of biomolecules. *Bioinformatics*, 25(16):2110–2117.

Fober, T., Mernberger, M., Klebe, G., and Hüllermeier, E. (2012). Fingerprint kernels for protein structure comparison. *Molecular Informatics*, 31(6-7):443–452.

Fober, T., Mernberger, M., Melnikov, V., Moritz, R., and Hüllermeier, E. (2009c). Extension and empirical comparison of graph-kernels for the analysis of protein active sites. In *Workshop Knowledge Discovery and Machine Learning*, pages 30–36, Darmstadt, Germany.

Fober, T., Mernberger, M., Moritz, R., and Hüllermeier, E. (2009d). Graph-kernels for the comparative analysis of protein active sites. In *German Conference on Bioinformatics*, pages 21 – 31, Halle (Saale), Germany.

Fodor, J. and Yager, R. (2002). *Fuzzy set-theoretic operators and quantifiers*. Kluwer, Dordrecht.

Fröhlich, H., , Wegner, J. K., Sieker, F., and Zell, A. (2005). Optimal assignment kernels for attributed molecular graphs. In *International conference on Machine learning*, pages 225 – 232, Bonn, Germany.

Gärtner, T. (2003). A survey of kernels for structured data. *SIGKKD Explorations*, 5(1):49 – 58.

Gärtner, T. (2008). *Kernels for structured data*. World Scientific, Singapore.

Gerstein, M. and Levitt, M. (1996). Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of protein structures. In *International Conference on Intelligent Systems for Molecular Biology*, volume 4, pages 59–67.

Gerstein, M. and Levitt, M. (1998). Comprehensive Assessment of Automatic Structural Alignment Against a Manual s´Standard, the Scop Classification of Proteins. *Protein Science*, 7(2):445–456.

Gibrat, J. F., Madej, T., and Bryant, S. H. (1996). Surprising similarities in structure comparison. *Current Opinion in Structural Biology*, 6(3):377–385.

Goodrich, M. T., Mitchell, J. S. B., and Orletsky, M. W. (1994). Practical methods for approximate geometric pattern matching under rigid motions. In *Annual Symposium on Computational Geometry*, pages 103 – 112, Stony Brook, New York, United States.

Grindley, H., Artymiuk, P., Rice, D., and Willett, P. (1993). Identification of Tertiary Structure Resemblance in Proteins using a Maximal Common Subgraph Isomorphism Algorithm. *Journal of Molecular Biology*, 229(3):707–721.

Hajek, P. (1998). *Metamathematics of fuzzy logic*. Kluwer, Dordrecht.

Harary, F. (1994). *Graph Theory*. Westview Press, Boulder, USA.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Man and Cybernetics*, 4(2):100–107.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical report, University of California at Santa Cruz.

Havel, T. F., Kuntz, I. D., and Crippen, G. M. (1983). The theory and practice of distance geometry. *Bulletin of Mathematical Biology*, 45(5):665–720.

Hazan, E., Safra, S., and Schwartz, O. (2003). On the complexity of approximating *k*-dimensional matching. In *Approximation, Randomization, and Combinatorial Optimization... Algorithms and Techniques*, volume 2764, pages 59–70. Springer.

Heffernan, P. J. and Schirra, S. (1992). Approximate decison algorithm for point set congruence. In *8th Annual ACM Symposium on Computational Geometry*, pages 93–101.

Hendlich, M., Bergner, A., Günther, J., and Klebe, G. (2003). Relibase: Design and development of a database for comprehensive analysis of protein-ligand interactions. *Journal of Molecular Biology*, 326:607–620.

Hendlich, M., Rippmann, F., and Barnickel, G. (1997). LIGSITE: Automatic and efficient detection of potential small molecule-binding sites in proteins. *Journal of Molecular Graphics and Modelling*, 15:359–363.

Hoffmann, B., Zaslavskiy, M., Vert, J.-P., and Stoven, V. (2010). A new protein binding pocket similarity measure based on comparison of clouds of atoms in 3d: application to ligand prediction. *BM*, 11(1):99–115.

Höfle, B., Geist, T., Rutzinger, M., and Pfeifer, N. (2007). Glacier surface segmentation using airborne laser scanning point cloud and intensity data. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Espoo, Finland.

Holder, L., Cook, D., and Djoko, S. (1994). Substructure Discovery in the Subdue System. In *AAAI Workshop on Knowledge Discovery in Databases*, pages 169–180.

Holm, L. and Park, J. (2000). DaliLite Workbench for Protein Structure Comparison. *Bioinformatics*, 16(6):566–567.

Holm, L. and Sander, C. (1993). Protein Structure Comparison by Alignment of Distance Matrices. *Journal of Molecular Biology*, 233(1):123–138.

Hopcroft, J. E. and Wong, J. K. (1974). Linear time algorithm for isomorphism of planar graphs. In Constable, R. L., Ritchie, R. W., Carlyle, J. W., and Harrison, M. A., editors, *Sixth annual ACM symposium on Theory of computing*, pages 172–184, Seattle, Washington. ACM.

Huan, J., Wang, W., Prins, J., et al. (2003). Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism. In *Third IEEE International Conference on Data Mining*, pages 549–561. IEEE Computer Society Washington, DC, USA.

Hüllermeier, E., Fober, T., and Mernberger, M. (2013). Fuzzy logik. In Dubitzky, W., Wolkenhauer, O., Cho, K.-H., and Yokota, H., editors, *Encyclopedia of Systems Biology*. Springer.

Huson, D. H. and Bryant, D. (2006). Application of phylogenetic networks in evolutionary studies. *Molecular Biology and Evolution*, 23(2):254–267.

Huttenlocher, D. P., Kedem, K., and Sharir, M. (1993a). The upper envelope of voronoi surfaces and its applications. *Discrete and Computational Geometry*, 9(1):267–291.

Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. J. (1993b). Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–864.

Imai, K., Sumino, S., and Imai, H. (1989). Minimax geometric fitting of two corresponding sets of points. In *5th Annual ACM Symposium on Computational Geometry*, pages 266–275.

Iman, R. L. (2008). Latin hypercube sampling. In *Encyclopedia of Quantitative Risk Analysis and Assessment*, New York, USA. Wiley.

Inokuchi, A., Washio, T., and Motoda, H. (2000). An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In *Fourth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23. Springer-Verlag London, UK.

Irfanoglu, M. O., Gökberk, B., and Akarun, L. (2004). 3d shape-based face recognition using automatically registered facial surfaces. In *17th International Conference on Pattern Recognition*.

Jambon, M., Imberty, A., Deleage, G., and Geourjon, C. (2003). A New Bioinformatic Approach to Detect Common 3 D Sites in Protein Structures. *Proteins Structure Function and Genetics*, 52(2):137–145.

Justice, D. and Hero, A. (2006). A Binary Linear Programming Formulation of the Graph Edit Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1200–1214.

Kabsch, W. (1976). A solution of the best rotation to relate two sets of vectors. *Acta Crystallographica*, 32:922–923.

Kanehisa, M., Goto, S., Kawashima, S., Okuno, Y., and Hattori, M. (2004). The KEGG resource for deciphering the genome. *Nucleic Acids Research*, 32:D277 – D280.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85 – 103, New York, USA. Plenum Press.

Karypis, G. (2006). CLUTO - family of data clustering software tools v 2.1.1. http://glaros.dtc.umn.edu/gkhome/views/cluto.

Kashima, H., Tsuda, K., and Inokuchi, A. (2003). Marginalized Kernels Between Labeled Graphs. In *20th International Conference on Machine Learning*, pages 321–328.

Kawabata, T. and Nishikawa, K. (2000). Protein Structure Comparison Using the Markov Transition Model of Evolution. *Proteins*, 41(1):108–122.

Kinoshita, K., Murakami, Y., and Nakamura, H. (2007). eF-seek: prediction of the functional sites of proteins by searching for similar electrostatic potential and molecular surface shape. *Nucleic Acid Research*, 35(suppl 2):W398–W402.

Kinoshita, K. and Nakamura, H. (2003). Identification of Protein Biochemical Functions by Similarity Search using the Molecular Surface Database eF-site. *Protein Science*, 12(8):1589–1595.

Kinoshita, K. and Nakamura, H. (2005). Identification of the Ligand Binding Sites on the Molecular Surface of Proteins. *Protein Science*, 14(3):711–718.

Klement, E., Mesiar, R., and Pap, E. (2002). *Triangular norms*. Kluwer, Dordrecht.

Kleywegt, G. and Jones, T. (1997). Detecting Folding Motifs and Similarities in Protein Structures. *Methods in Enzymology*, 277:525–545.

Kondor, R. and Borgwardt, K. M. (2008). The skew spectrum of graphs. In *International Conference on Machine Learning*, pages 496–503.

Kondor, R. and Lafferty, J. (2002). Diffusion Kernels on Graphs and Other Discrete Structures. In *19th International Conference on Machine Learning*, pages 315–322.

Krotzky, T. (2011). *Analyse und algorithmische Erweiterung von Methoden zum strukturellen Vergleich von Proteinbindestellen*. Grin Verlag, München, Germany.

Kuhn, D. (2004). *Beschreibung von Proteinbindetaschen für Funktionsstudien und de Novo-Design und die Entwicklung von Methoden zur funktionellen Klassifizierung von Proteinfamilien*. PhD thesis, Philipps-Universität Marburg, Marburg, Germany.

Kuhn, D., Weskamp, N., Schmitt, S., Hüllermeier, E., and Klebe, G. (2006). From the Similarity Analysis of Protein Cavities to the Functional Classification of Protein Families using Cavbase. *Journal of Molecular Biology*, 359(4):1023–1044.

Kuhn, H. (2005). The Hungarian Method for the Assignment Problem. *Naval Research Logistics*, 52(1):7–21.

Kupas, K., Ultsch, A., and Klebe, G. (2007). Large scale analysis of protein-binding cavities using self-organizing maps and wavelet-based surface patches to describe functional properties, selectivity discrimination, and putative cross-reactivity. *Proteins*, 71:1288–1306.

Kuramochi, M. and Karypis, G. (2001). Frequent Subgraph Discovery. In *IEEE International Conference on Data Mining*, pages 313–320.

Kuramochi, M. and Karypis, G. (2007). Discovering Frequent Geometric Subgraphs. *Information Systems*, 32(8):1101–1120.

Lamdan, Y. and Wolfson, H. (1988). Geometric Hashing: A General And Efficient Model-based Recognition Scheme. In *Second International Conference on Computer Vision*, pages 238–249.

Leibowitz, N., Fligelman, Z. Y., Nussinov, R., and Wolfson, H. J. (1999). Multiple structural alignment and core detection by geometric hashing. In *International Conference on Intelligent Systems for Molecular Biology*, pages 169–177, Heidelberg, Germany.

Leibowitz, N., Nussinov, R., and Wolfson, H. (2001). MUSTA-A General, Efficient, Automated Method for Multiple Structure Alignment and Detection of Common Motifs: Application to Proteins. *Journal of Computational Biology*, 8(2):93–121.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Levi, G. (1973). A Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs. *Calcolo*, 9(1972):341–352.

Li, X.-L., Tan, S.-H., Foo, C.-S., and Ng, S.-K. (2005). Interaction graph mining for protein complexes using local clique merging. *Genome Informatics*, 16(2):260–269.

Liu, G. and Wong, L. (2008). Effective pruning techniques for mining quasi-cliques. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, part II*, pages 33–49, Antwerp, Belgium.

Lladós, J., Martí, E., and Villanueva, J. (2001). Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *Pattern Analysis and Machine Intelligence*, 23(10):1137–1143.

Madej, T., Gibrat, J., and Bryant, S. (1995). Threading a Database of Protein Cores. *Proteins*, 23(3):356–369.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

McGregor, J. J. (1982). Backtrack search algorithms and the maximal common subgraph problem. *Software - Practice and Experience*, 12(1):23–34.

Mernberger, M., Klebe, G., and Hüllermeier, E. (2011). SEGA - A semi-global approach to graph alignment for approximate molecular structure comparison. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, PrePrints.

Messmer, B. and Bunke, H. (1998a). A new Algorithm for Error-tolerant Subgraph Isomorphism Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504.

Messmer, B. and Bunke, H. (1998b). Error-Correcting Graph Isomorphism using Decision Trees. *International Journal of Pattern Recognition and Artificial Intelligence*, 12:721–742.

Mitchell, E., Artymiuk, P., Rice, D., and Willett, P. (1990). Use of Techniques Derived from Graph Theory to Compare Secondary Structure Motifs in Proteins. *Journal of Molecular Biology*, 212(1):151–166.

Mizuguchi, K. and Go, N. (1995). Comparison of Spatial Arrangements of Secondary Structural Elements in Proteins. *Protein Engineering Design and Selection*, 8(4):353–362.

Mollineda, R., Vidal, E., and Casacuberta, F. (2002). A windowed weighted approach for approxiate cyclic string matching. In Kasturi, R., Laurendeau, D., and Suen, C., editors, *16th International Conference on Pattern Recognition*, pages 188–191, Quebec, Canada.

Munoz, D., Vandapel, N., and Hebert, M. (2008). Directional associative markov network for 3-d point cloud classification. In *Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, Paris, France.

Munshi, A. (2011). *The OpenCL Specification*. Khronos OpenCL Working Group, Beaverton, Oregon, USA.

Myers, R., Wilson, R., and Hancock, E. (2000). Bayesian Graph Edit Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):628–635.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal Molecular Biology*, 48(3):443–453.

Neuhaus, M. and Bunke, H. (2004). A Probabilistic Approach to Learning Costs for Graph Edit Distance. In *17th International Conference on Pattern Recognition*, volume 3, pages 389–393.

Neuhaus, M. and Bunke, H. (2005). Self-organizing Maps for Learning the Edit Costs in Graph Matching. *IEEE Transactions on Systems, Man and Cybernetics*, B35(3):503–514.

Neuhaus, M. and Bunke, H. (2006). A Convolution Edit Kernel for Error-tolerant Graph Matching. In *18th International Conference on Pattern Recognition*, volume 4, pages 220–223.

Neuhaus, M. and Bunke, H. (2007a). Automatic learning of cost functions for graph edit distance. *Information Sciences*, 177(1):239–247.

Neuhaus, M. and Bunke, H. (2007b). *Briding the Gap between Graph Edit Distance and Kernel Machines*. World Scientific, New Jersey.

Nocedal, J. and Wright, S. J. (2000). *Numerical Optimization*. Springer, Berlin, Germany.

Norvig, P. (1991). *Paradigms of Artificial Intelligence Programming*. Morgan Kaufmann, Burlington, Massachusetts, USA.

Okada, K. and Ling, H. (2007). An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):840–853.

Orengo, C. and Taylor, W. (1996). SSAP: Sequential Structure Alignment Program for Protein Structure Comparison. *Methods in Enzymology*, 266:617–35.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project.

Papadopoulos, A. and Manolopoulos, Y. (1999). Structure-based Similarity Search with Graph Histograms. In *10th International Workshop on Database and Expert Systems Applications*, pages 174–178.

Pedrycz, W. and Gomide, F. (2007). *Fuzzy systems engineering: toward human-centric computing*. Wiley, New York.

Pelillo, M. (1998). A Unifying Framework for Relational Structure Matching. In *14th International Conference on Pattern Recognition*, volume 2.

Peris, G. and Marzal, A. (2002). Fast cyclic edit distance computation with weighted edit costs in classification. In Kasturi, R., Laurendeau, D., and Suen, C., editors, *16th International Conference on Pattern Recognition*, volume 4, pages 184–187, Quebec, Canada.

Pfeffer, P., Fober, T., Hüllermeier, E., and Klebe, G. (2009). Garlig: A fully automated tool for the subset selection of large fragment spaces via a self-adaptive genetic algorithm or the comparative evaluation of 5 different scoring functions used for de novo design. *Journal of Chemical Information and Modelling*, 50(9):1644–1659.

Powers, R., Copeland, J. C., Germer, K., Mercier, K. A., Ramanathan, V., and Revesz, P. (2006). Comparison of protein active site structures for functional annotation of proteins and drug design. *PROTEINS: Structure, Function, and Bioinformatics*, 65:124–135.

Pérot, S., Sperandio, O., Miteva, M. A., Camproux, A.-C., and Villoutreix, B. O. (2010). Druggable pockets and binding site centric chemical space: a paradigm shift in drug discovry. *Drug Discovery Today*, 15(15 / 16):656–667.

Ralaivola, L., Swamidass, S., Saigo, H., and Baldi, P. (2005). Graph Kernels for Chemical Informatics. *Neural Networks*, 18(8):1093–1110.

Raymond, J., Gardiner, E., and Willett, P. (2002). Heuristics for Similarity Searching of Chemical Graphs Using a Maximum Common Edge Subgraph Algorithm. *Jorunal of Chemical Information and Computer Sciences*, 42(2):305–316.

Raymond, J. and Willett, P. (2002). Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, 16(7):521–533.

Read, R. and Corneil, D. (1977). The Graph Isomorphism Disease. *Journal of Graph Theory*, 1(1):339–363.

Riesen, K. and Bunke, H. (2010). *Graph Classification and Clustering based on Vector Space Embedding*. World Scientific, London, UK.

Robles-Kelly, A. and Hancock, E. (2005). Graph Edit Distance from Spectral Seriation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):365–378.

Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121.

Russell, B. (1923). Vagueness. *The Australasian Journal of Psychology and Philosophy*, 1:84–92.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423.

Sander, O., Sing, T., Sommer, I., Low, A. J., Cheung, P. K., Harrigan, P. R., Lengauer, T., and Domingues, F. S. (2007). Structural Descriptors of gp120 V3 Loop for the Prediction of HIV-1 Coreceptor Usage. *PLoS Computational Biology*, 3(3):555–564.

Sanfeliu, A. and Fu, K. (1983). A Distance Measure Between Attributed Relational Graphs for Pattern Recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13(3):353–362.

Schmidt, D. and Druffel, L. (1976). A Fast Backtracking Algorithm to Test Directed Graphs for Isomorphism Using Distance Matrices. *Journal of the ACM*, 23(3):433–445.

Schmitt, S., Kuhn, D., and Klebe, G. (2002). A new method to detect related function among proteins independent of sequence and fold homology. *Journal of Molecular Biology*, 323(2):387–406.

Schwefel, H.-P. (1993). *Evolution and Optimum Seeking*. John Wiley & Sons, Inc., New York, USA.

Shasha, D., Wang, J., and Giugno, R. (2002). Algorithmics and Applications of Tree and Graph Searching. In *Proceedings. 21th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 39–52. ACM Press New York, NY, USA.

Shatsky, M. (2006). *The Common Point Set Problem with Applications to Protein Structure Analysis*. PhD thesis, Tel Aviv University, Tel Aviv, Israel.

Shatsky, M., Niussinov, R., and Wolfson, H. J. (2004). A method for simultaneous alignment of multiple protein structures. *Proteins: Structure, Function, and Bioinformatics*, 56:143–156.

Shatsky, M., Shulman-Peleg, A., Nussinov, R., and Wolfson, H. J. (2006). The multiple common point set problem and its application to molecule binding pattern detection. *Journal of Computational Biology*, 13(2):407–428.

Shawe-Taylor, J. and Cristianini, N. (2003). *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambrigde.

Shindyalov, I. and Bourne, P. (1998). Protein Structure Alignment by Incremental Combinatorial Extension (CE) of the Optimal Path. *Protein Engineering Design and Selection*, 11(9):739–747.

Shindyalov, I. and Bourne, P. (2001). A Database and Tools for 3-D Protein Structure Comparison and Alignment using the Combinatorial Extension (CE) Algorithm. *Nucleic Acids Research*, 29(1):228–229.

Shulman-Peleg, A., Nussinov, R., and Wolfson, H. (2004). Recognition of Functional Sites in Protein Structures. *Journal of Molecular Biology*, 339(3):607–633.

Siggelkow, S. and Burkhardt, H. (2002). Improvement of histogram-based image retrieval and classification. In *16th International Conference on Pattern Recognition*, volume 3, page 30367.

Singh, A. and Brutlag, D. (1997). Hierarchical protein Structure Superposition Using Both Secondary Structure and Atomic Representations. In *International Conference on Intelligence Systems for Molecular Biology*, volume 5, pages 284–293.

Sokal, R. R. and Michener, C. D. (1958). A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 38:1409–1438.

Spriggs, R., Artymiuk, P., and Willett, P. (2003). Searching for Patterns of Amino Acids in 3D Protein Structures. *Journal of Chemical Information and Computer Sciences*, 43(2):412–421.

Sprinzak, J. and Werman, M. (1994). Affine point matching. *Pattern Recognition Letters*, 15:337–339.

Srinivasan, A., King, R., Muggleton, S., and Sternberg, M. (1997). Carcinogenesis Predictions Using ILP. In *7th International Workshop on Inductive Logic Programming*, pages 273–287. Springer-Verlag London, UK.

Suganthan, P. N., Teoh, E., and Mital, D. (1995). Pattern recognition by graph matching using the potts MFT neural networks. *Pattern Recognition*, 28(7):997–1009.

Taylor, W., Flores, T., and Orengo, C. (1994). Multiple Protein Structure Alignment. *Protein Science*, 3(10):1858–1870.

Thompson, W. B., Owen, J. C., de St. Germain, H. J., S. R. Stark, J., and Henderson, T. C. (1999). Feature-based reverse engineering of mechanical parts. *IEEE Transactions on robotics and automation*, 15(1):57–66.

Tian, Y., McEachin, R. C., Santos, C., States, D. J., and Patel, J. M. (2007). SAGA: A subgraph matching tool for biological graphs. *Bioinformatics*, 23(2):232–239.

Tian, Y. and Patel, J. M. (2008). TALE: A tool for approximate large graph matching. In *International Conference on Data Engineering*, pages 963–972, Cancun, Mexico.

Tomita, E., Tanaka, A., and Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363:28–42.

Tsai, W. and Fu, K. (1979). Error-correcting isomorphism of attributed relational graphs for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 9(12):757–768.

Ullmann, J. (1976). An Algorithm for Subgraph Isomorphism. *Journal of the ACM*, 23(1):31–42.

Umeyama, S. (1988). An Eigendecomposition Approach to Weighted Graph Matching Problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):695–703.

Vajda, S. and Guarnieri, F. (2006). Characterization of protein-ligand interaction sites using experimental and computational methods. *Current Opinion in Drug Discovery and Development*, 9:354–362.

Vert, J.-P. (2008). The optimal assignment kernel is not positive definite. Technical report, Centre for Computational Biology, Mines Paris Tech.

Vertan, C. and Boujemaa, N. (2000). Using fuzzy histograms and distances for color image retrieval. In *Challenge of Image Retrieval*, pages 1–6, Brighton, United Kingdom.

Vishwanathan, S., Borgwardt, K. M., Kondor, I. R., and Schraudolph, N. N. (2008). Graph kernels. *Journal of Machine Learning Research*, 9:1–41.

Vriend, G. and Sander, C. (1991). Detection of Common Three-dimensional Substructures in Proteins. *Proteins: Structure, Function and Genetics*, 11(1):52–58.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM*, 21(1):168 – 173.

Wang, J., Zhang, K., and Chirn, G. (1994a). Approximate Graph Matching using Probabilistic Hill Climbing algorithms. In *6th International Conference on Tools with Artificial Intelligence*, pages 390–396.

Wang, J., Zhang, K., and Chirn, G. (1994b). The Approximate Graph Matching Problem. In *12th IAPR International Conference on Computer Vision & Image Processing*, volume 2, pages 284–288.

Wang, X. and Wang, J. (2000). Fast Similarity Search in Three-Dimensional Structure Databases. *Journal of Chemical Information and Computer Sciences*, 40(2):442–451.

Wang, Y., Fan, K., and Horng, J. (1997). Genetic-based Search for Error-correcting Graph Isomorphism. *IEEE Transactions on Systems, Man and Cybernetics*, 27(4):588–597.

Wassermann, S. and Faust, K. (1994). *Social Network Analysis: Methods and Applications*. Cambridge University Press.

Watson, J. D., Laskowski, R. A., and Thornton, J. M. (2005). Predicting protein function from sequence and structural data. *Current Opinion in Structural Biology*, 15(3):275–284.

Weisel, M., Proschak, E., Kriegl, J. M., and Schneider, G. (2009). Form follows function: Shape analysis of protein cavities for receptor-based drug design. *PROTEOMICS*, 9(2):451–549.

Werman, M., Peleg, S., and Rosenfeld, A. (1985). A distance metric for multi-dimensional histograms. *Computer, Vision, Graphics, and Image Processing*, 32:328–336.

Weskamp, N., Hüllermeier, E., and Klebe, G. (2009). Merging chemical and biological space: Structural mapping of enzyme binding pocket space. *Proteins: Structure Function and Bioinformatics*, 76:317–330.

Weskamp, N., Hüllermeier, E., Kuhn, D., and Klebe, G. (2007). Multiple graph alignment for the structural analysis of protein active sites. *IEEE Transactions on Computational Biology and Bioinformatics*, 4(2):310–320.

Wheeler, T. J. and Kececioglu, J. D. (2007). Multiple alignment by aligning alignments. *Bioinformatics*, 23(13):i559–i568.

Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining*. Morgan Kaufmann, Burlington, Massachusetts, USA.

Wolfson, H. J. and Rigoutsos, I. (1997). Geometric hashing: An overview. IEEE Computational Science and Engineering.

Xenarios, I., Salwinski, L., Duan, X., Higney, P., Kim, S., and Eisenberg, D. (2002). DIP, the database for interacting proteins: A research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, 30:303–305.

Xu, L. and Oja, E. (1990). Improved Simulated Annealing, Boltzmann Machine, and Attributed Graph Matching. In *EURASIP Workshop on Neural Networks*, pages 151–160. Springer-Verlag London, UK.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183 – 190.

Yan, X. and Han, J. (2003). CloseGraph: Mining Closed Frequent Graph Patterns. In *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 286–295. ACM Press New York, NY, USA.

Yan, X., Yu, P., and Han, J. (2004). Graph Indexing: A Frequent Structure-based Approach. In *ACM SIGMOD International Conference on Management of Data*, pages 335–346. ACM New York, NY, USA.

Yan, X., Yu, P., and Han, J. (2005). Substructure Similarity Search in Graph Databases. In *ACM SIGMOD International Conference on Management of Data*, pages 766–777. ACM Press New York, NY, USA.

Yan, X., Zhu, F., Han, J., and Yu, P. (2006). Searching Substructures with Superimposed Distance. In *International Conference on Data Engineering*, volume 88.

Yoshida, K. and Motoda, H. (1995). CLIP: Concept Learning from Inference Patterns. *Artificial Intelligence*, 75(1):63–92.

Zadeh, L. (1983). A computational approach to fuzzy quantifiers in natural languages. *Computing and Mathematics with Applications*, 9:149–184.

Zeng, Z., Wang, J., and Zhou, L. (2007). Out-of-core coherent closed quasi-clique mining from large dense graph databases. *ACM Transactions on Database Systems*, 32(2):Article 13.

Zhang, K., Wang, J., and Shasha, D. (1995). On the Editing Distance Between Undirected Acyclic Graphs and related problems. *Combinatorial Pattern Matching*, 937(1):395–407.

Zhang, S., Hu, M., and Yang, J. (2007). Treepi: A novel graph indexing method. In *23th International Conference on Data Engineering*, pages 966–975.

A

Results on Graph-based Approaches

Table A.1: Classification rates on the ATP/NADH dataset obtained by using local clique merging ($\gamma = 0.6$) for different values of k and λ.

λ \ k	1	3	5	7	9
0.0	0.583	0.592	0.594	0.586	0.586
0.1	0.592	0.594	0.600	0.586	0.594
0.2	0.594	0.600	0.597	0.583	0.597
0.3	0.594	0.597	0.597	0.586	0.603
0.4	0.606	0.603	0.603	0.589	0.606
0.5	0.620	0.611	0.611	0.586	0.606
0.6	0.628	0.617	0.620	0.594	0.608
0.7	0.645	0.625	0.625	0.594	0.608
0.8	0.668	0.631	0.623	0.597	0.606
0.9	**0.687**	0.639	0.634	0.611	0.594
1.0	0.682	0.628	0.623	0.606	0.597

Table A.2: Classification rates on the ATP/NADH dataset obtained by using local clique merging ($\gamma = 0.7$) for different values of k and λ.

λ \ k	1	3	5	7	9
0.0	0.738	0.715	0.682	0.659	0.654
0.1	0.744	0.721	0.690	0.676	0.670
0.2	0.746	0.724	0.699	0.679	0.685
0.3	0.752	0.730	0.715	0.690	0.690
0.4	0.761	0.746	0.735	0.701	0.699
0.5	0.797	0.777	0.761	0.724	0.730
0.6	0.839	0.797	0.786	0.755	0.766
0.7	0.845	0.823	0.797	0.775	0.780
0.8	0.854	0.851	0.834	0.803	0.803
0.9	**0.862**	0.842	0.839	0.808	0.823
1.0	**0.862**	0.837	0.837	0.814	0.808

Table A.3: Classification rates on the ATP/NADH dataset obtained by using local clique merging ($\gamma = 0.8$) for different values of k and λ.

λ \ k	1	3	5	7	9
0.0	0.727	0.690	0.676	0.656	0.673
0.1	0.732	0.696	0.682	0.665	0.682
0.2	0.732	0.710	0.679	0.670	0.685
0.3	0.755	0.713	0.696	0.679	0.693
0.4	0.772	0.732	0.701	0.699	0.713
0.5	0.794	0.755	0.718	0.710	0.724
0.6	0.817	0.792	0.746	0.741	0.738
0.7	0.837	0.800	0.772	0.758	0.749
0.8	**0.859**	0.820	0.817	0.792	0.777
0.9	0.856	0.831	0.834	0.811	0.820
1.0	0.839	0.825	0.811	0.808	0.797

Table A.4: Classification rates on the ATP/NADH dataset obtained by using local clique merging ($\gamma = 0.9$) for different values of k and λ.

λ \ k	1	3	5	7	9
0.0	0.713	0.693	0.659	0.648	0.637
0.1	0.721	0.707	0.670	0.662	0.642
0.2	0.735	0.724	0.676	0.685	0.654
0.3	0.741	0.735	0.687	0.693	0.673
0.4	0.752	0.738	0.713	0.707	0.696
0.5	0.786	0.758	0.735	0.721	0.707
0.6	0.808	0.766	0.752	0.744	0.738
0.7	0.820	0.789	0.777	0.775	0.761
0.8	0.834	0.828	0.800	0.780	0.786
0.9	0.839	0.839	0.834	0.817	0.792
1.0	**0.851**	0.825	0.820	0.811	0.794

Table A.5: Classification rates on the ATP/NADH dataset obtained by using the Bron-Kerbosch algorithm for different values of k and λ.

λ \ k	1	3	5	7	9
0.0	0.817	0.823	0.825	0.828	0.820
0.1	0.825	0.820	0.837	0.831	0.823
0.2	0.825	0.825	0.837	0.831	0.817
0.3	0.828	0.823	0.831	0.831	0.825
0.4	0.837	0.828	0.831	0.831	0.828
0.5	0.837	0.839	0.839	0.831	0.825
0.6	0.842	0.834	0.837	0.831	0.825
0.7	0.839	0.828	0.831	0.834	0.825
0.8	0.851	0.837	0.823	0.820	0.823
0.9	**0.865**	0.854	0.839	0.831	0.820
1.0	**0.865**	0.851	0.839	0.831	0.831

Table A.6: Classification rates on the ATP/NADH dataset obtained by using the local clique approach for different values of k and λ.

λ \ k	1	3	5	7	9
0.0	0.715	0.685	0.642	0.614	0.611
0.1	0.713	0.690	0.645	0.617	0.623
0.2	0.715	0.690	0.654	0.631	0.639
0.3	0.724	0.693	0.662	0.637	0.654
0.4	0.732	0.690	0.673	0.656	0.668
0.5	0.752	0.704	0.690	0.676	0.685
0.6	0.769	0.732	0.707	0.701	0.701
0.7	0.786	0.763	0.741	0.735	0.713
0.8	0.820	0.803	0.786	0.775	0.769
0.9	0.828	0.828	0.817	0.811	0.803
1.0	**0.848**	0.820	0.817	0.800	0.780

B

Datasets

Table B.1: CavBase IDs of the ATP set

1a0i.4	1b76.4	1b8a.4	1c0g.4	1csn.1	1dej.6
1dv2.3	1dy3.1	1e2q.1	1e4g.1	1e8x.10	1esq.4
1esv.3	1f9a.10	1fmw.3	1g21.18	1g64.2	1gn8.1
1gol.3	1gtr.6	1gz3.7	1gz4.18	1h1v.4	1h1w.1
1hi1.6	1hlu.2	1i7l.4	1ijj.3	1j09.1	1j1z.10
1j7k.2	1jwa.2	1kax.2	1kay.1	1kaz.1	1kh2.6
1kj8.2	1kj9.2	1kp2.2	1kp3.2	1kvk.4	1kxp.7
1l2t.4	1lhr.7	1m83.3	1ma9.8	1mau.2	1miw.1
1n56.5	1n75.1	1nge.1	1ngf.1	1ngg.1	1ngh.1
1nsf.2	1nyr.9	1o93.4	1o9t.5	1obg.1	1ojl.10
1p8z.4	1pk8.22	1px2.6	1q97.8	1qhx.1	1qmz.7
1qrs.4	1qrt.3	1qru.4	1r8b.1	1rgi.7	1rys.4
1s9j.2	1tqp.1	1tyq.18	1u5r.1	1uev.1	1v1b.6
1w7a.12	1x01.2	1xdn.2	1xdp.14	1xef.5	1y8q.12
1yid.9	1yun.2	1zao.1	1zyd.3	2a3z.3	2a40.9
2a41.4	2a42.2	2aqx.2	2aru.1	2bek.1	2bup.1
2c96.2	2c9c.2	2cch.6	2cci.8	2cjm.8	2ddo.2
2dra.2	2dto.3	2dxt.3	2eww.1	2f02.4	2faq.4
2fgj.9	2gnk.1	2gwj.1	2hix.3	2hmp.3	2hmw.1

2i4o.8	2idx.6	2iyw.1	2j9c.3	2j9e.1	2npi.10
2ogx.5	2olq.1	2p55.2	2p9k.14	2p9s.15	2pbd.1
2phk.1	2pze.4	2pzf.2	2q0d.7	2q31.2	2q66.1
2q7g.1	2qb8.5	2qk4.3	2r6x.1	2r7l.1	2vhq.7
3c4w.10	3c4x.9	3cjc.6			

Table B.2: CavBase IDs of the NADH set

1a9y.2	1a9z.1	1ahh.2	1ahi.2	1b14.3	1b8u.1
1bdb.1	1bmd.1	1bpw.4	1bw9.3	1bxg.7	1bxk.4
1c1d.2	1c1x.3	1cdo.1	1cer.5	1cw3.24	1cwu.5
1d1s.5	1d1t.5	1d4f.15	1d7o.2	1dbv.10	1deh.5
1dhr.1	1dhs.3	1dir.4	1dli.3	1dqs.6	1e3s.3
1e3w.4	1ee2.3	1efl.8	1ej2.1	1ek5.1	1eno.2
1eny.2	1enz.2	1ez4.9	1fdv.12	1fk8.4	1fmc.2
1gad.1	1gae.2	1geg.1	1geu.2	1giq.3	1gt2.3
1hdr.1	1hdx.5	1hdy.6	1hdz.6	1hex.1	1hku.1
1hl3.2	1hld.4	1hlp.4	1ht0.2	1htb.5	1hwy.10
1hzj.2	1i3k.2	1i3l.2	1i3n.2	1ib0.4	1ie3.5
1j0x.4	1j5p.3	1jq5.1	1ju9.5	1jvf.3	1jw7.3
1k0u.12	1k4m.5	1kvq.1	1kvr.1	1kvu.1	1kyq.1
1ldg.3	1ldy.4	1lj8.2	1lrj.2	1lrk.2	1lrl.2
1lsj.5	1lvl.2	1m76.3	1m8f.1	1m8g.1	1m8j.1
1m9h.1	1ma0.4	1mc5.5	1mg0.7	1ml3.7	1mp0.4
1mx3.2	1nah.1	1nai.2	1nff.2	1nfr.3	1nr5.5
1o6z.5	1o9j.17	1obb.6	1oc4.5	1og3.1	1ojs.2
1p45.4	1pj3.17	1psd.5	1qs2.4	1rlz.3	1roz.1
1rqd.4	1sby.3	1sg6.2	1t24.3	1t2d.3	1tae.9
1teh.1	1u5c.2	1u7h.1	1u8f.4	1uda.1	1udb.1
1udc.1	1uwk.5	1uwl.5	1uxj.3	1uxk.2	1uxt.2

1v59.5	1vbi.2	1vc2.1	1vi2.3	1vjp.4	1vjt.3
1vko.6	1vm6.4	1w1u.5	1wze.9	1x14.6	1x1t.1
1x7d.1	1x87.5	1xag.1	1xah.4	1xaj.3	1xal.3
1xel.1	1xwf.7	1yba.24	1yc2.4	1yl7.16	1ywg.4
1z2i.8	1z45.5	1zbq.14	1znq.7	1zrq.5.	2a9k.1
2b36.10	2b69.1	2bkj.2	2c20.4	2c54.4	2c59.4
2c5a.3	2c5e.3	2d37.1	2d8a.1	2dc1.5	2dld.2
2dt5.1	2dvm.7	2ed4.1	2eer.2	2ep7.1	2fnz.7
2fr8.4	2fzw.4	2g76.4	2gag.11	2gah.11	2gwl.1
2h7l.2	2h7m.2	2hae.3	2hu2.2	2i9p.2	2ixa.1
2ixb.1	2nad.10	2o2s.7	2ome.11	2oxi.5	2pd3.3
2pd6.5	2pzm.2	2q1t.1	2q1u.1	2q1w.8	2qjo.8
2udp.2	3b6j.1	3bts.7	3dbv.10	3had.5	3hdh.9
3hud.3	4mdh.3	5mdh.3	6adh.5		

Erklärung

Ich versichere, dass ich meine Dissertation „Geometric, Feature-based and Graph-based Approaches for the Structural Analysis of Protein Binding Sites: Novel Methods and Computational Analysis“ selbständig, ohne unerlaubte Hilfe angefertigt und mich dabei keiner anderen als der von mir ausdrücklich bezeichneten Quellen und Hilfen bedient habe. Die Dissertation wurde in der jetzigen oder einer ähnlichen Form noch bei keiner anderen Hochschule eingereicht und hat noch keinen sonstigen Prüfungszwecken gedient.

Marburg, den 26.02.2013

Thomas Fober

Sachsenring 16
35041 Marburg
Germany

thomas@mathematik.uni-marburg.de
http://www.uni-marburg.de/fb12/kebi/people/thomas

Education

Since October 2007: PhD-student in Computer Science at the Department Mathematics and Computer Science, Philipps-Universität Marburg, Germany.

October 2001 — March 2007: Student in Computer Science (minor Economics) at Universität Dortmund, Germany. Degree: Diplom (German Master)

1997 — 2000: Gymnasiale Oberstufe, Iserlohn, Germany (German high school)

1991 — 1997: Realschule, Iserlohn, Germany (German secondary school)

1987 — 1991: Grundschule, Iserlohn, Germany (German primary school)

Research Stays

January 16^{th} — 28^{th}, 2012 : The Cambridge Crystallographic Data Centre, Cambridge, United Kingdom.

Organization of Conferences

June 28^{th} — July 2^{nd}, 2010: 13^{th} *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Westfalenhallen, Dortmund, Germany

September 17^{th} — September 19^{th}, 2012: 6^{th} *International Conference on Scalable Uncertainty Management*, Marburg, Germany

Reviewer Activities

International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.

IEEE / ACM Transactions on Computational Biology and Bioinformatics.

ACS Journal of Chemical Information and Modeling.

Soft Computing (Springer Verlag).

Journal Articles

Thomas Fober, Marco Mernberger, Gerhard Klebe, Eyke Hüllermeier: Evolutionary Construction of Multiple Graph Alignments for the Structural Analysis of Biomolecules. *Bioinformatics*, 25(16): 2110–2117, 2009.

Yu Yi, Thomas Fober, Eyke Hüllermeier: Fuzzy Operator Trees for Modeling Rating Functions. *International Journal of Computational Intelligence and Applications*, 8(4), 413–428, 2009.

Patrick Pfeffer, Thomas Fober, Eyke Hüllermeier, Gerhard Klebe: GARLig: A fully Automated Tool for Subset Selection of Large Fragment Spaces via a Self-Adaptive Genetic Algorithm. *Journal of Chemical Information and Modeling*, 50(9), 1644–1659, 2010.

Thomas Fober, Serghei Glinca, Gerhard Klebe, Eyke Hüllermeier: Superposition and Alignment of Labeled Point Clouds. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(6), 1653–1666, 2011.

Thomas Fober, Marco Mernberger, Gerhard Klebe, Eyke Hüllermeier: Fingerprint Kernels for Protein Structure Comparison. *Molecular Informatics*, 31(6-7): 443–452, 2012.

Robin Senge, Thomas Fober, Maryam Nasiri, Eyke Hüllermeier: Fuzzy Pattern Trees — Ein alternativer Ansatz zu Fuzzy-Modellierung. *at — Automatisierungstechnik*, 60(10): 622–629, 2012.

Michiel Stock, Thomas Fober, Eyke Hüllermeier, Serghei Glinca, Gerhard Klebe, Tapio Pahikkala, Antti Airola, Bernard De Baets, Willem Waegeman: Identification of functionally-related enzymes by learning-to-rank methods and cavity-based similarity measures. *IEEE/ACM Transactions on Computational*

Biology and Bioinformatics, submitted.

Thomas Fober, Marco Mernberger, Eyke Hüllermeier: Graph-based methods for protein structure comparison. *Wiley Interdisciplinary Reviews*, submitted.

Peer-Reviewed Conferences

Thomas Fober, Eyke Hüllermeier, Marco Mernberger: Evolutionary Construction of Multiple Graph Alignments for the Structural Analysis of Biomolecules. In *German Conference on Bioinformatics*, Dresden, Germany, 2008.

Thomas Fober, Eyke Hüllermeier, Marco Mernberger: Evolutionary Construction of Multiple Graph Alignments for Mining Structured Biomolecular Data. In *Workshop Knowledge Discovery and Machine Learning*, Würzburg, Germany, 2008.

Thomas Fober, Eyke Hüllermeier: Fuzzy Modeling of Labeled Point Cloud Superposition for the Comparison of Protein Binding Sites. In *IFSA/EUSFLAT World Conference*, Lisboa, Portugal, 2009.

Thomas Fober, Marco Mernberger, Vitalik Melnikov, Ralph Moritz, Eyke Hüllermeier: Extension and Empirical Comparison of Graph-Kernels for the Analysis of Protein Active Sites. In *Workshop Knowledge Discovery and Machine Learning*, Darmstadt, Germany, 2009.

Thomas Fober, Marco Mernberger, Ralph Moritz, Eyke Hüllermeier: Graph-Kernels for the Comparative Analysis of Protein Active Sites. In *German Conference on Bioinformatics*, Halle (Saale), Germany, 2009.

Thomas Fober, Gerhard Klebe, Eyke Hüllermeier: Efficient Construction of Multiple Geometrical Alignments for the Comparison of Protein Binding Sites. In *IEEE International Conference on Intelligent Systems Design and Applications*, Pisa, Italy, 2009.

Imen Boukhris, Zied Elouedi, Thomas Fober, Marco Mernberger, Eyke Hüllermeier: Similarity Analysis of Protein Binding Sites: A Generalization of the Maximum Common Subgraph Measure based on Quasi-Clique Detection. In *IEEE International Conference on Intelligent Systems Design and Applications*, Pisa, Italy, 2009.

Thomas Fober, Eyke Hüllermeier: Similarity Measures for Protein Structures based on Fuzzy Histogram Comparison. In *IEEE World Congress on Computational Intelligence*, Barcelona, Spain, 2010.

Thomas Fober, Marco Mernberger, Gerhard Klebe, Eyke Hüllermeier: Efficient Similarity Retrieval for Protein Binding Sites based on Histogram Comparison. In *German Conference on Bioinformatics*, Braunschweig, Germany, 2010.

Thomas Fober, Gerhard Klebe, Eyke Hüllermeier: Local Clique Merging: An Extension of the Maximum Common Subgraph Measure for the Classification of Graph Structures. In *Joint Conference of the German Classification Society and the German Association for Pattern Recognition*, Frankfurt a. M., Germany, 2011.

Matthias Leinweber, Lars Baumgärtner, Marco Mernberger, Thomas Fober, Eyke Hüllermeier, Gerhard Klebe, Bernd Freisleben: GPU-based Cloud Computing for Comparing the Structure of Protein Binding Sites. In *IEEE Conference on Digital Ecosystem Technologies – Complex Environment Engineering*, Campione d'Italia, Italy, 2012.

Other Publications

Thomas Fober: Experimentelle Analyse Evolutionärer Algorithmen auf dem CEC 2005 Testfunktionensatz. Technical report TR06-2-009, Universität Dortmund, Germany, 2006.

Thomas Fober, Marco Mernberger, Eyke Hüllermeier: Evolutionary Construction of Multiple Graph Alignments for the Structural Analysis of Biomolecules. In *17th Workshop Computational Intelligence*, Witten-Bommerholz, Germany, 2007.

Thomas Fober: Evolutionary Methods for Protein Structure Comparison. *Romanian-German Symposium on Mathematics and Its Applications*, Sibiu, Romania, May, 2009.

Thomas Fober, Eyke Hüllermeier: Multiple Geometrical Alignments for the Analysis of Protein Active Sites. In *19th Workshop Computational Intelligence*, Witten-Bommerholz, Germany, 2009.

Thomas Fober, Weiwei Cheng, Eyke Hüllermeier: Focusing Search in Multi-

objective Evolutionary Optimization through Preference Learning from User Feedback. In *21th Workshop Computational Intelligence*, Dortmund, Germany, 2011.

Eyke Hüllermeier, Sebastian Link, Thomas Fober, Bernhard Seeger (Eds): Scalable Uncertainty Management. Springer, Heidelberg, 2012.

Eyke Hüllermeier, Thomas Fober, Marco Mernberger: Classification. In: Dubitzky W., Wolkenhauer O., Cho K.-H., Yokota H. (Eds): *Encyclopedia of Systems Biology*. Springer, New York, USA, (to appear).

Eyke Hüllermeier, Thomas Fober, Marco Mernberger: Fuzzy logic. In: Dubitzky W., Wolkenhauer O., Cho K.-H., Yokota H. (Eds): *Encyclopedia of Systems Biology*. Springer, New York, USA, (to appear).

Eyke Hüllermeier, Thomas Fober, Marco Mernberger: Case-based reasoning. In: Dubitzky W., Wolkenhauer O., Cho K.-H., Yokota H. (Eds): *Encyclopedia of Systems Biology*. Springer, New York, (to appear).

Teaching Assistant

Summer 2007:	Fuzzy Systems
Winter 2007 / 08:	Logic and Discrete Mathematik
Summer 2008:	Fuzzy Systems Knowledge Engineering & Bioinformatics Seminar
Summer 2009:	Computational Intelligence
Winter 2009 / 10:	Logic and Discrete Mathematik
Summer 2010:	Computational Intelligence
Winter 2010 / 11:	Introduction to Bioinformatics Efficient Algorithms
Summer 2012:	Fuzzy Systems

Supervised Diploma-, Master- and Bachelor Theses

2008: Kernbasierte Ähnlichkeitsmaße auf Graphen by Ralph Moritz

2009: Graph-based similarity of protein binding sites derived from quasi-clique detection by Imen Boukhris

2010: Analyse und algorithmische Erweiterung von Methoden zum strukturellen Vergleich von Proteinbindestellen by Timo Krotzky

2011: Effiziente Nächste-Nachbar Suche in dynamischen geometrischen Strukturen by Adil Paul

2011: Entwurf und Analyse von Methoden zur Berechnung maximaler Cliquen im Produkt zweier euklidischer Graphen by Mabou Bopda Constant

Awards

Young Author Award 2007 from the Gesellschaft für Informatik (GI) and Verein Deutscher Ingenieure (VDI) for the outstanding contribution at the 17th Workshop Computational Intelligence.

Young Author Award 2009 from the Verein Deutscher Ingenieure (VDI) for the outstanding contribution at the 19th Workshop Computational Intelligence.